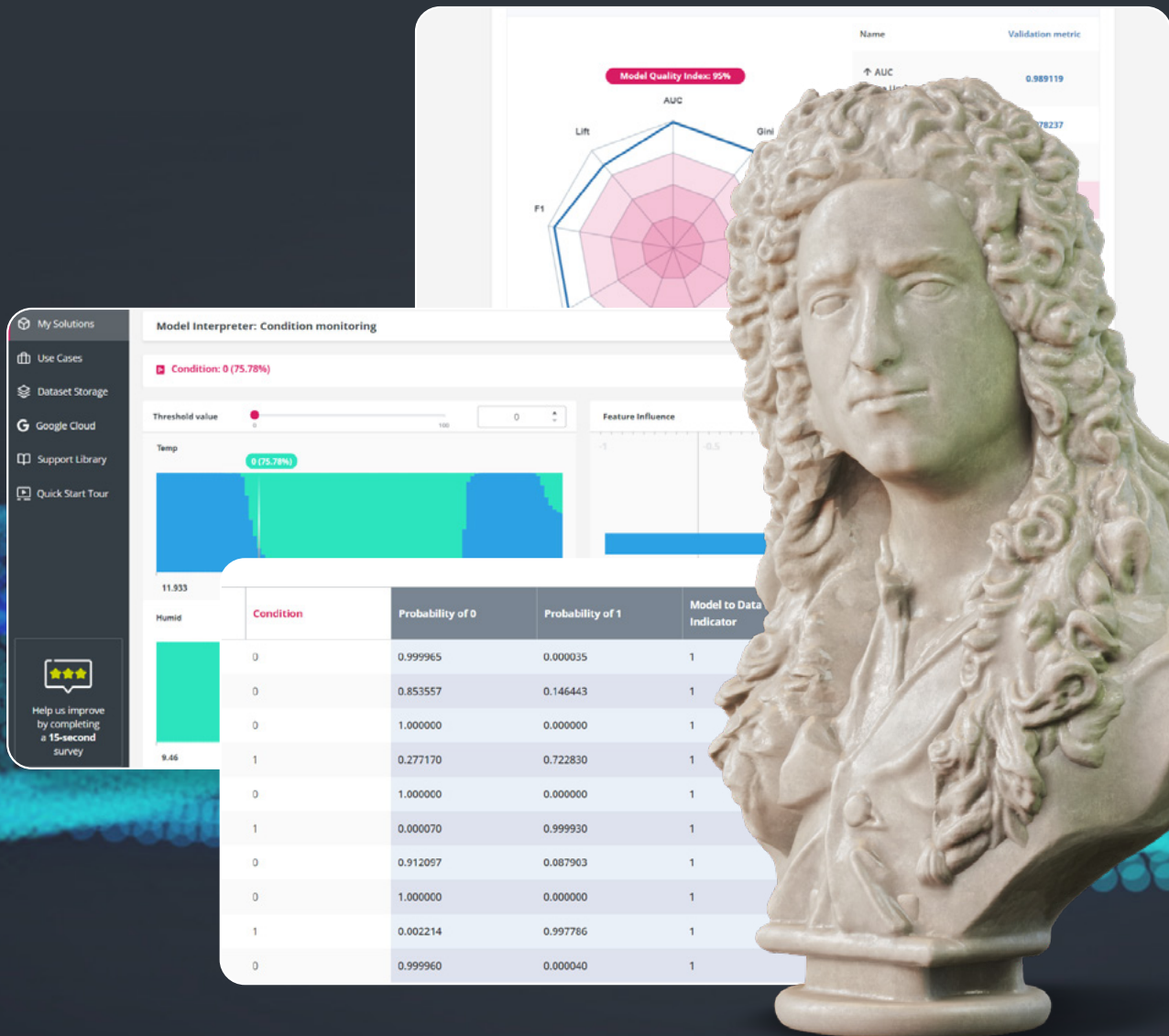




Intelligent Agent
Neutron



Neutron User Guide

CONTENT

Terms and Definitions	5
About Neuton TinyML Platform	11
Workflow Overview	14
Dataset Requirements for Sensor and Tabular Data	15
Requirements of requirements for training datasets	15
Requirements of requirements for testing datasets (or new data)	16
How to identify and change file encoding	17
1. Select Data for Training	19
Create a New Solution	19
Select Data (Dataset tab)	20
Specify Dataset Options	23
2. Train your Model	24
Training parameters	24
Start and control the Model training process	31
Metrics	33
Analytics Tools	34
Log	35
3. Make Predictions	36
Info Area	36
C Library	37
Web Prediction	40
REST API Access	43
Disabling Predictions	44
Audio File Processing	45

Explainability Office	50
Exploratory Data Analysis	50
Model Quality Index	52
Model Quality Diagram	53
Feature Importance Matrix	54
Model Interpreter	55
Solutions Management	57
Dataset Storage	60
Dataset Management	62
Metrics Definition	63
Regression Metrics	63
Mean Absolute Error (MAE) and related Metrics	63
Root Mean Squared Error (RMSE)	63
Root Mean Squared Logarithmic Error (RMSLE)	63
Coefficient of Determination (R2)	63
Mean Squared Error (MSE)	64
Root Mean Squared Percentage Error (RMSPE)	64
Binary Classification Metrics	64
Accuracy	64
Balanced Accuracy	64
Precision	64
Recall	65
F1 Score	65
Confusion Matrix	65
Gini	65
AUC (ROC AUC)	65

Lift	66
LogLoss	66
Multiclass Classification Metrics	66
Accuracy.....	66
Balanced Accuracy	66
Macro Average Precision.....	67
Weighted Average Precision	67
Macro Average Recall	67
Weighted Average Recall.....	67
Macro Average F1 Score.....	67
Weighted Average F1 Score.....	68
Confusion Matrix	68
LogLoss	68

TERMS AND DEFINITIONS

This section provides a glossary of terms and definitions used in this guide. We do not intend to give scientific definitions, but we want to provide general terms and descriptions which are clear for users with or without a data science background.

A

Artificial Intelligence

Artificial Intelligence (AI) is an area of computer science that emphasizes the creation of intelligent machines that work and react like humans.

C

Classification

Classification (classification task) is the prediction of a target variable represented as a range of discrete classes. Binary classification tasks are represented by a target variable with two possible classes. Multiclass classification tasks are represented by a target variable with 3 or more classes.

Confidence Interval

Confidence Interval is applicable for regression task types. With XX% probability, shows the possible prediction spread for each predicted value.

D

Data Preprocessing

Data Preprocessing is a process of dataset preparation for model training, including data cleaning, missing values imputation, removal of outliers and variables transformation, etc.

Dataset

Dataset is a volume of data (statistics) in tabular format.

E

Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA) is a set of charts displaying the main characteristics of a training dataset.

F

Feature

Feature (independent variable, predictor) is represented by a column in a dataset that characterizes the target variable.

Feature Extraction

Feature Extraction is an extraction of additional information (creation of the new variables) from existing data.

Feature Importance Matrix (FIM)

Feature Importance Matrix (FIM) is a chart which represents the 10 features that had the most significant impact on the model prediction of the target variable.

Frequency functions

- FFT first peak power
- FFT first peak frequency
- FFT second peak power
- FFT second peak frequency
- FFT third peak power
- FFT third peak frequency

These functions use fast Fourier transform to calculate the frequency of peaks and their power. If Frequency features are selected the window size should be equal to the power of 2.

G

Global peak to peak of high frequency

Global peak to peak of high frequency (amplitude function) is a calculating of the high-frequency signal by subtracting the moving average filter output from the original signal (feature). You can specify the Smoothing Factor which is the amount of attenuation for frequencies over the cutoff frequency. The smoothing factor value should be equal to the power of 2 but not greater than the window size.

Global peak to peak of low frequency

Global peak to peak of low frequency (amplitude function) is a calculating of the low-frequency signal by applying a moving average filter with a smoothing factor. The smoothing factor value should be equal to the power of 2 but not greater than the window size.

K

Kurtosis

Kurtosis is a statistical measure of the combined weight of a distribution's tails relative to the center of the distribution.

L

Lag Feature

A lag feature is a name for a variable which contains data from prior time steps.

M

Machine Learning

Machine Learning (ML) is the scientific study of algorithms and statistical models that computer systems use to perform a specific task without using explicit instructions, relying on patterns and inference instead.

Max

Max is a statistical function which calculates the maximum of the column (feature).

Mean

Mean is a statistical function which calculates the arithmetic mean of the column (feature).

Mean crossings

This statistical function calculates the number of times the selected column crosses the mean.

Metadata

Metadata is the information about the data

Metric

Metric is a functional value which describes model quality.

Min

Min is a statistical function which calculates the minimum of the column (feature).

Model

Model is a mathematical representation of dependencies between the features (independent variables) and the target variable.

Model Interpreter

Model Interpreter allows you to interactively change the values of original features and see prediction results in real time. For continuous and discrete features, the Model Interpreter builds a graphical representation of their relation to the target variable, and for classification tasks, you can see the probabilities of predicted classes on the graph. It also allows you to specify the threshold value and see the feature values for which prediction results are below or above the threshold. Furthermore, feature influence for continuous features will show you the trend in the target variable value.

Model-to-Data Relevance Indicator

Taking into account feature impact on the model prediction, the Model-to-Data Relevance Indicator calculates the statistical similarity between the data uploaded for predictions and the data used for model training. The measure (100-M2D) shows the possible degradation level for the target metric value.

A low value for M2D may indicate a significant change in the model input data (data drift) that can lead to model performance degradation (model decay).

For example, if the Model-to-Data Relevance Indicator on the current dataset (uploaded for predictions) equals 95%, then the user can reasonably expect the model quality to decrease by as much as 5% from the validation metric.

100% = no change in model input data, no model performance degradation.

1% = a significant change in model input data, that leads to significant model performance degradation.

Model-to-Data Relevance Indicator is calculated for:

- every row sent for predictions.
- dataset sent for predictions
- all data sent for predictions aggregated over time (Historical Model-to-Data Relevance Indicator)

Model Quality Diagram

Model Quality Diagram is a graphical representation of model quality in relation to metric indicator values that are scaled in the range [0-1], where 1 is the ideal quality of the model, and 0 is the minimum quality of the model.

Model Quality Index

Determines the quality of the model based on the metric indicator values.

Model Quality Index Value Range: 1 - 100%

Minimum quality: 1%

Maximum quality: 100%

The Model Quality Index is calculated during the model training based on the training, or the validation dataset (if the User uploaded the validation dataset for model training). The correlation between the Training Model Quality Index and the acceptable model predictive power depends significantly on the problem being solved by the model.

Thus for tasks that do not require high model predictive power, the acceptable range of the Model Quality Index values is 75-100%, while for high-precision tasks it is 99-100%.

N

Negative mean crossings

Computes the number of times the selected input crosses the mean with a negative slope.

Neural Network

A Neural Network is a series of algorithms that endeavors to recognize underlying relationships in a dataset through a process which is close to the way the human brain operates.

P

Petrosian fractal dimension

This statistical function converts the data to a binary sequence and estimates the fractal dimension from time series.

Positive mean crossings

Computes the number of times the selected input crosses the mean with a positive slope.

Prediction /Inference

Prediction is the output of a model after it has been trained and applied to new data, when one is trying to predict unknown values of the target variable.

R

Regression

Regression (regression task type) is predicting a continuous value (for example predicting the prices of a house given the house features like location, size, number of bedrooms, etc).

Root mean square

The root mean square is the root of the arithmetic mean of the squares of a set of numbers.

S

Sensor Data

Sensor data is the data from gyroscopes, accelerometers, magnetometers, electromyography (EMG), and other similar devices.

Skewness

The skewness statistical function measures the asymmetry of the distribution of a variable.

Solution

Solution is an object in Neuton in which all model parameters are specified. All workflow actions are executed inside the solution.

Splitting

Splitting is the process of separating a dataset into two parts one for training and one for validation.

T

Tabular Data

Tabular data is the data used for solving AutoML tasks (not suitable for TinyML tasks).

Target Variable

Target Variable is a variable the model is learning to predict. Target variable may be represented as a range of discrete classes or as continuous real numbers.

TinyML

TinyML is a field of study in Machine Learning and Embedded Systems that explores the types of models you can run on small, low-powered devices like microcontrollers. It enables low-latency, low-power, and low-bandwidth model inference on edge devices.

Total Footprint

Total Footprint is the amount of space in FLASH memory and SRAM which the model uses for prediction.

Training

Training is the process of "learning" to uncover relationships between the features of a particular dataset and the target variable.

Training Dataset

A training Dataset is the input dataset (or its part) which the machine learning algorithm uses to "learn" to uncover relationships between its features and the target variable.

V

Validation

Validation is the quality assessment process for a model which has been trained and built to predict on a particular target variable.

Validation Dataset

The validation dataset is another subset of the input data used to predict the target variable with the trained model, and measure the error between the known target values in the validation dataset and the predictions.

Validation Metrics

Metric is a functional value which describes model quality applying to the holdout validation dataset or cross-validation process.

ABOUT NEUTON TINYML PLATFORM

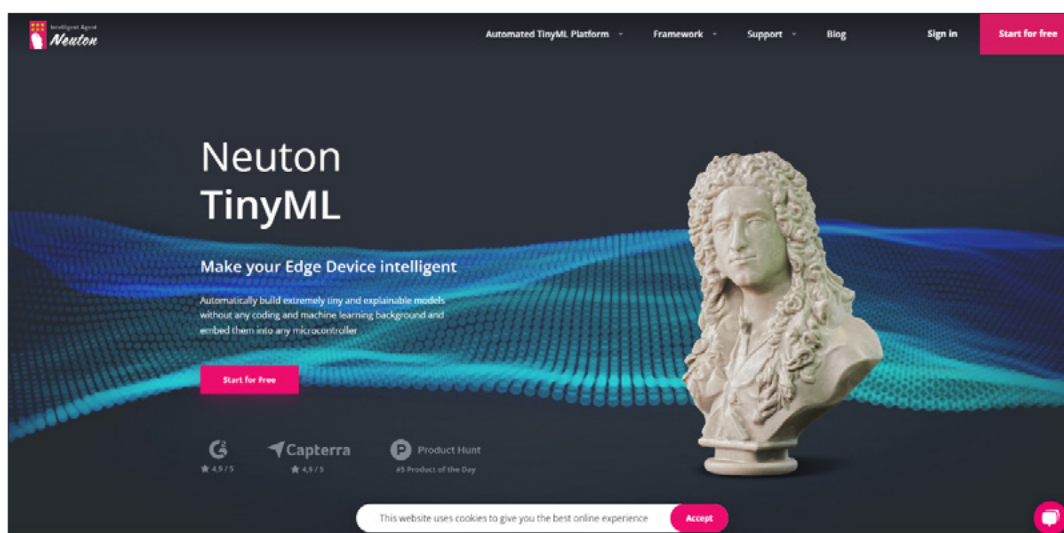
Neuton is a Neural Network Framework that was created by our scientists and engineers from scratch. It is not based on any derivative or existing frameworks, or non-neural algorithms. The resulting models built with Neuton are compact, fast, self-growing and learning. Furthermore, Neuton is so easy to use that no special AI background is needed.

Neuton effectively builds tiny and accurate models without overfitting and compression techniques while leveraging significantly less data than other competing algorithms. Subsequently, the resulting models are very compact with exceptionally fast predictions, exceeding those of all contemporary competitors.

Neuton Auto ML solution is based on Neuton's Neural Network Framework and automates the whole process of creating models for edge devices through the use of innovative and cutting-edge technology.

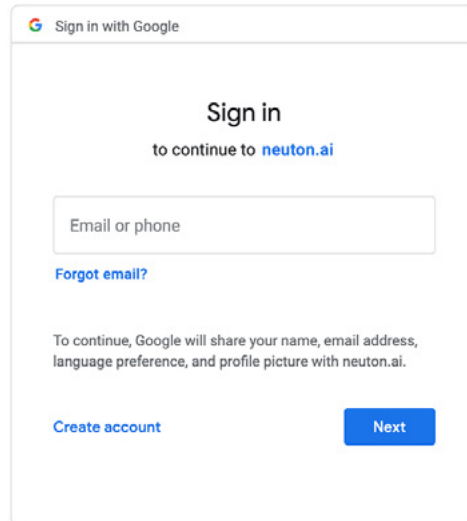
Neuton can be accessed via an Internet browser. Neuton has different subscription options. The test-drive subscription plan is free, only preloaded datasets are available for training. The test-drive version is available with any Google account. In a Zero Gravity plan you pay only for the infrastructure used (for Training Time, Predict Time, and for Storage), the Neuton service is free. You can use your own datasets for model training, a subscription via Google Cloud Platform is required. The third option is an Enterprise plan. The plan also includes extensive professional data science services provided by Neuton.ai experts, that guarantee the successful implementation of the first use case! Neuton provides assistance in building the first model, interpreting the results, implementing it into the business process, and managing the life cycle of the model, and also help with embedding the model into the device.

To access Neuton, type the landing page address <https://neuton.ai> in your browser navigation bar.



Picture 1 - Neuton Landing page

Click the **"Sign in"** button to start using Neutron. You will be redirected to the **"Sign in with Google"** web page. Once there, provide your email address and the password used during the Neutron registration process.

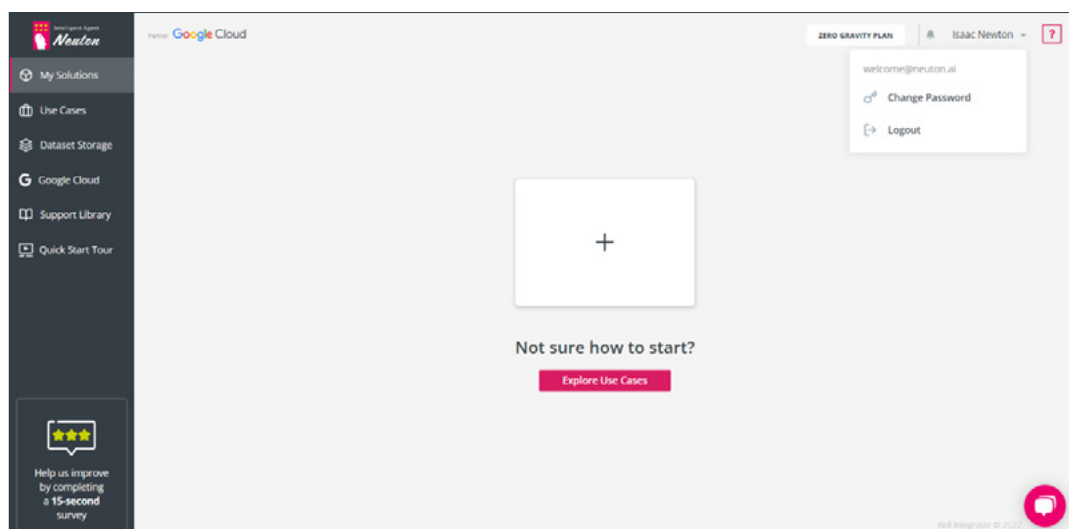


Picture 2 - Sign in with Google page

NOTE:

If you are already signed in with the Google account linked to Neutron, you can use the following URL: <https://login.neutron.ai>

If you would like to log out or to log in with another account, click on your username in the top right corner and then click **"Logout"**:



Picture 3 - Neutron Platform default view

The Neuton platform also provides a test-drive version which requires no registration. In the test-drive version, you can see how various use cases are solved as well as download pretrained models. Plus, you can train a model yourself using preloaded datasets, which may give you inspiration on how to solve your own cases.

To build your own models set up a Google Cloud Platform account and subscribe to Neuton's **Free Zero Gravity plan**. Use our service absolutely free of charge, covering only your Google Cloud Platform infrastructure costs.

WORKFLOW OVERVIEW

The Neuton platform enables users to use the system to predict various outcomes based on statistics. This is made possible by processing the user data and using it to train a machine learning model.

A typical dataset for solving tasks on sensor and tabular data contains independent features usually referred to as “predictors” and dependent variables which the model is learning to predict, referred to as “target” or target variable. Usually, the dataset is represented in a tabular format where each row of data represents the feature values separated by a comma. The first row in the table is used for column names.

Training is performed on the training dataset imported by the user. Model validation presumes training the model on the training dataset and predicting on the validation dataset. Validation is performed either on the split portion of the training dataset or on an independent validation dataset imported by the user (if applicable).

During training, the model is looking for patterns and dependencies between the predictors and the target variable.

After the model has been trained, the user can use it to predict on new data via various options provided by the platform.

The full machine learning workflow and pipeline within Neuton Auto ML consists of **3 simple steps:**

- 1. Select data for training**
- 2. Train your model**
- 3. Make predictions**

Dataset Requirements for Sensor and Tabular Data

In this section the requirements are provided for training and test datasets. Meeting these requirements will guarantee successful model training and prediction on new data. We recommend using a text editor like Notepad++ to check the datasets.

Requirements of requirements for training datasets

1. Dataset must be a CSV file using UTF-8 or ISO-8859-1 encoding.
2. Dataset must have a minimum of 2 columns, 50 rows and headers.
3. All feature values in the dataset must be numeric, the target column also can be represented by string values for classification task type. For regression task types the target variable must have only numeric values.
4. Dataset must not have any empty values or values which represent empty values like "NA", "NAN" and etc.
5. The first row in the dataset must contain the column names, and a comma, semicolon, pipe, caret or tab must be used as a separator. CRLF or LF should be used as the end-of-line character. The separator and end-of-line character should be consistent inside the dataset.
6. All column names (values in the csv file header) must be unique and must contain only letters (a-z, A-Z), numbers (0-9), hyphens (-) or underscores (_).
7. For the classification task type, a training dataset must have minimum 2 classes of target variable with at least 10 samples provided for each class.
8. Currently Neuton supports only the EN-US locale for numbers, so:
 - You must use a dot as a decimal separator, and delete spaces and commas typically used to separate every third digit in your numeric fields.
 - Example – "20,000.00" should be replaced with "20000.00"
 - If any numeric column is represented as a combination of a number and its corresponding unit, then only the number should be placed in the column.
 - Example – "\$20,000.00" should be replaced with "20000.00"
 - Date/time columns must be in epoch time representation or relative date format. For example, "10/18/2017" should be represented as "1508284800".
 - In test datasets the same timestamp format should be used in a manner consistent with the training dataset.
9. End-of-line symbols must be excluded from the field values.
10. In the case of sensor data from gyroscopes, accelerometers, magnetometers, electromyography (EMG), and other similar devices for creating models using Digital Signal Processing, every row of a dataset should be device readings

per unit of time with a label as a target. You should not shuffle signal labels or encode your signal for model creation.

```

1 ID,Date,Class,Name,Age,SibSp,Parch,Ticket,Fare,Cabin,Embarked,Target
2 1,3/14/12,3,"Braund, Mr. Owen Harris",22,1,0,A/5 21171,7.25,,S,0
3 2,3/15/12,1,"Cumings, Mrs. John Bradley (Florence Briggs Thayer)",38,1,0,PC 17599,71.2833,C85,C,1
4 3,3/16/12,3,"Heikkinen, Miss. Laina",26,0,0,STON/O2. 3101282,7.925,,S,1
5 4,3/17/12,1,"Futrelle, Mrs. Jacques Heath (Lily May Peel)",35,1,0,113803,53.1,C123,S,1
6 5,3/18/12,3,"Allen, Mr. William Henry",35,0,0,373450,8.05,,S,0
7 6,3/19/12,3,"Moran, Mr. James",,0,0,330877,8.4583,,0,0
8 7,3/20/12,1,"McCarthy, Mr. Timothy J",54,0,0,17463,51.8625,E46,S,0
9 8,3/21/12,3,"Palsson, Master. Gosta Leonard",2,3,1,349909,21.075,,S,0
10 9,3/22/12,3,"Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)",27,0,2,347742,11.1333,,S,1
11 10,3/23/12,2,"Nasser, Mrs. Nicholas (Adele Achem)",14,1,0,237736,30.0708,,C,1
12 11,3/24/12,3,"Sandstrom, Miss. Marguerite Rut",4,1,1,PP 9549,16.7,66,S,1
13 12,3/25/12,1,"Bonnell, Miss. Elizabeth",58,0,0,113783,26.55,C103,S,1
14 13,3/26/12,3,"Saunderscock, Mr. William Henry",20,0,0,A/5. 2151,8.05,,S,0
15 14,3/27/12,3,"Andersson, Mr. Anders Johan",39,1,5,347082,31.275,,S,0
16 15,3/28/12,3,"Vestrom, Miss. Hulda Amanda Adolfina",14,0,0,350406,7.8542,,S,0
17 16,3/29/12,2,"Hewlett, Mrs. (Mary D Kingcome)",,55,0,0,248786,16,,S,1
18 17,3/30/12,3,"Rice, Master. Eugene",2,4,1,382652,29.125,,0,0
19 18,3/31/12,2,"Williams, Mr. Charles Eugene",,0,0,244373,13,,S,1
20 19,4/1/12,3,"Vander Planke, Mrs. Julius (Emelia Maria Vandemoortele)",31,1,0,345763,18,,S,0
21 20,4/2/12,3,"Masselmani, Mrs. Fatima",,0,0,2649,7.225,,C,1
22 21,4/3/12,3,"Fynney, Mr. Joseph J",35,0,0,239865,26,,S,0
23 22,4/4/12,2,"Beesley, Mr. Lawrence",34,0,0,248698,13,056,S,1
24 23,4/5/12,3,"McGowan, Miss. Anna ""Annie""",15,0,0,330923,8.0292,,0,1
25 24,4/6/12,1,"Sloper, Mr. William Thompson",28,0,0,113780,35.5,A6,S,1
26 25,4/7/12,3,"Palsson, Miss. Torborg Danira",8,3,1,349909,21.075,,S,0
27 26,4/8/12,3,"Asplund, Mrs. Carl Oscar (Selma Augusta Emilia Johansson)",38,1,5,347077,31.3875,,S,1
28 27,4/9/12,3,"Emir, Mr. Farred Chehab",,0,0,2631,7.225,,C,0
29 28,4/10/12,1,"Fortune, Mr. Charles Alexander",19,3,2,19950,263,C23 C25 C27,S,0
30 29,4/11/12,3,"O'Dwyer, Miss. Ellen ""Nellie""",,0,0,330959,7.8792,,0,1
31 30,4/12/12,3,"Todoroff, Mr. Lalio",,0,0,349216,7.8958,,S,0
32 31,4/13/12,1,"Uruchurtu, Don. Manuel E",40,0,0,PC 17601,27.7208,,C,0
33 32,4/14/12,1,"Spencer, Mrs. William Augustus (Marie Eugenie)",,1,0,PC 17569,146.5208,878,C,1
34 33,4/15/12,3,"Glynn, Miss. Mary Agatha",,0,0,335677,7.75,,0,1
35 34,4/16/12,2,"Mheadon, Mr. Edward H",66,0,0,C.A. 24579,10.5,,S,0
36 35,4/17/12,1,"Meyer, Mr. Edgar Joseph",28,1,0,PC 17604,82.1788,,C,0
37 36,4/18/12,1,"Holverson, Mr. Alexander Oskar",42,1,0,113789,52,,S,0
38 37,4/19/12,3,"Mamee, Mr. Hanna",,0,0,2677,7.2292,,C,1
39 38,4/20/12,3,"Cann, Mr. Ernest Charles",21,0,0,A./S. 2152,8.05,,S,0
40 39,4/21/12,3,"Vander Planke, Miss. Augusta Maria",18,2,0,345764,18,,S,0
41 40,4/22/12,3,"Nicola-Yarred, Miss. Jamila",14,1,0,2651,11.2417,,C,1

```

Picture 4 - Training dataset example

Requirements of requirements for test datasets (or new data)

1. Dataset must be a CSV file using UTF-8 or ISO-8859-1 encoding.
2. For prediction on the platform (using a web interface) first row in the dataset must contain the column names, and a comma, semicolon, pipe, caret or tab must be used as a separator. CRLF or LF should be used as the end of a line character. When transferring data for inference on the device, the values must always be in the same order as in the training dataset (without target).
3. The test dataset must have the same file structure with the same requirements for the feature values as the training dataset. In case of removing some features in the platform web interface, the same features should be excluded from the test dataset.
4. The order of fields must be the same as in the training dataset. For example: if in the training dataset the order of columns is 'B, C, A'. Then the input data for prediction must be in the same order: value for feature B, after that - value for feature C and the next one - value for feature A.
5. End-of-line symbols must be excluded from the field values.

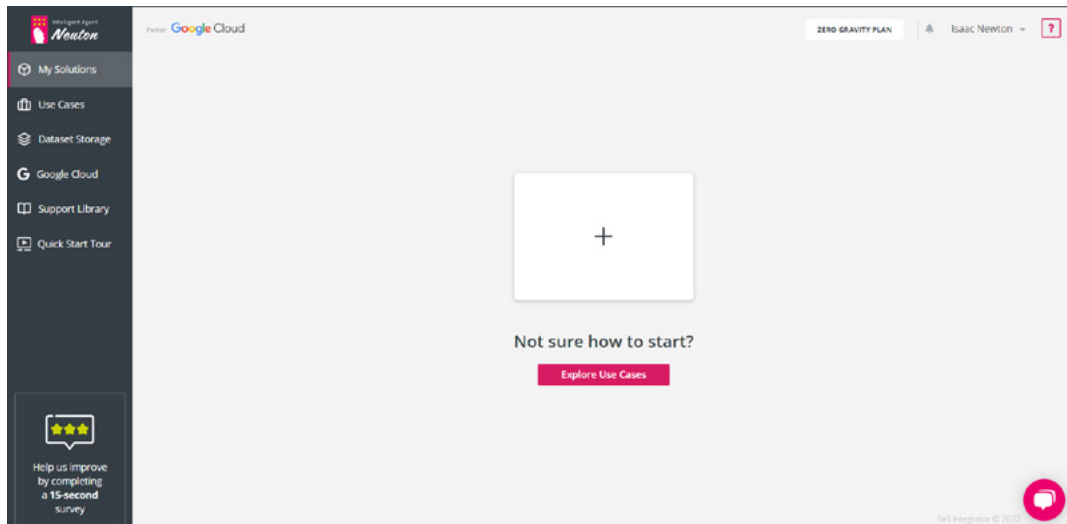
[illegible]

1. Select Data for Training

Create a new Solution

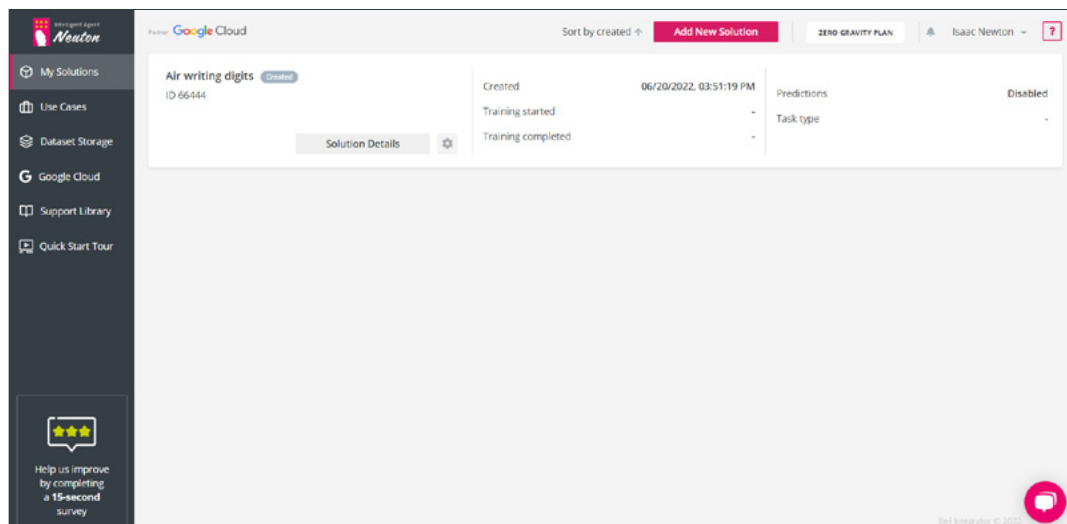
The model creation process within **Neuton** begins with creating a new **Solution**.

The solution is the object for training parameters specification and prediction processes management.



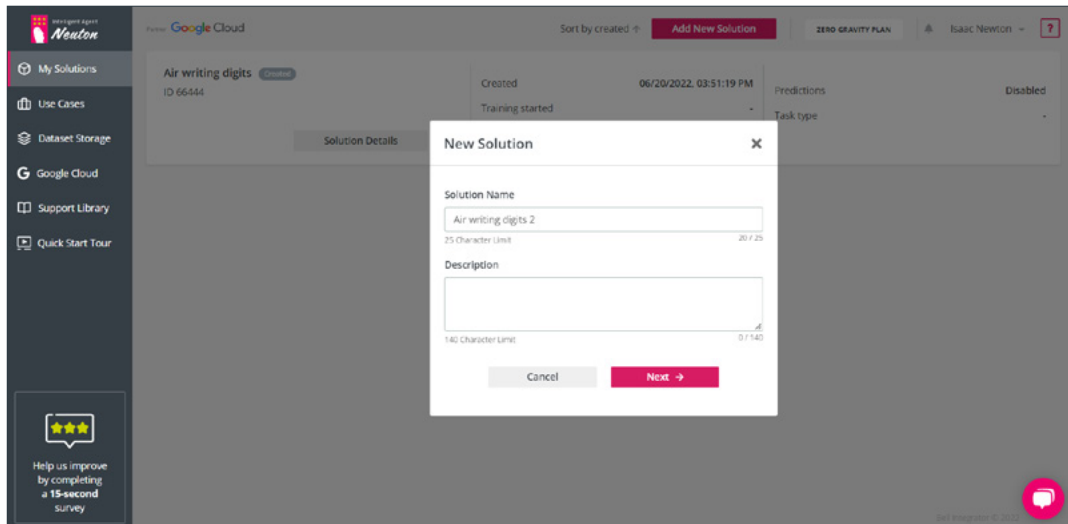
Picture 8 - My Solutions. Default view

After you have worked with Neuton and created some solutions, you can view your solutions in the **"My Solutions"** work space:



Picture 9 - My Solutions

To create a new solution with a list of previously created solutions click **"Add New Solution"**. The **"New Solution"** pop-up window will appear. Add the desired **"Solution Name"**.



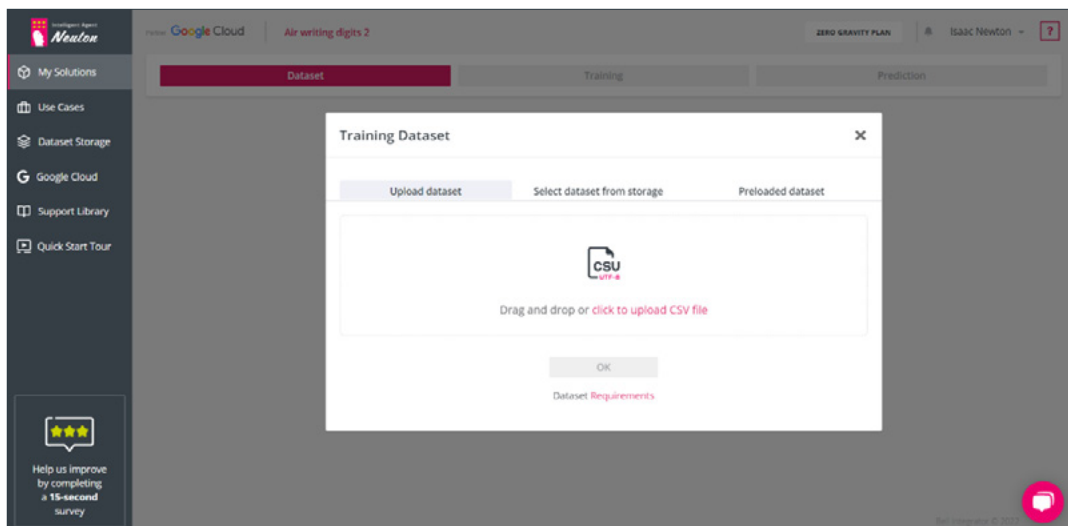
Picture 10 - New solution

After that, click the **"Next"** to go to dataset import/selection.

Select Data (Dataset tab)

Please read first about the requirements for training data in the **"Dataset requirements"** section.

Training dataset selection



Picture 11 - Dataset selection options

To select the dataset you have the following options (tabs):

- **Upload dataset**
This tab allows you to upload your own dataset. (This option is not available for the test-drive version.)
- **Select dataset from storage**
On this tab you can select one of your previously loaded datasets.

- **Preloaded dataset**

Neuton provides preloaded datasets (for demonstration and testing purposes).

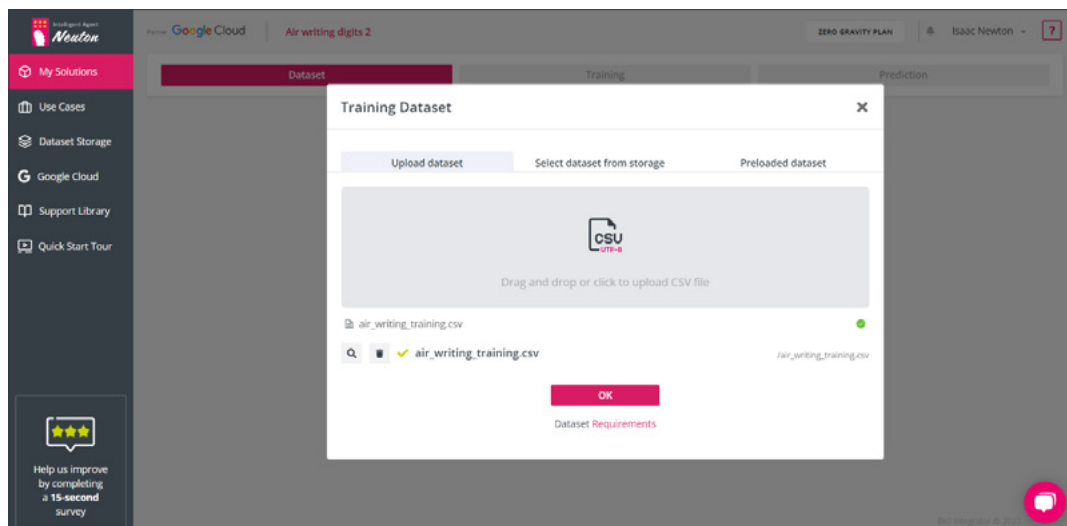
Select one of these options to specify the training dataset for your solution.

NOTE:

Upon uploading data to the platform, the data is encrypted in the cloud. All the resulting datasets and models are encrypted as well.

Upload Dataset

Please select "**Click to upload CSV file**" (see Picture above) and browse to the file location on your hard drive (or drag & drop):



Picture 12 - Upload Dataset

During the uploading process platform checks the dataset. If you see an error message you should verify the dataset and upload it again. When the file is successfully uploaded you will see a green **check mark**.

NOTE:

Machine learning operations cannot be performed on inappropriate data structures and variable types. Please make sure your dataset is processed accordingly. To read more about dataset requirements please refer to the "**Dataset requirements**" section

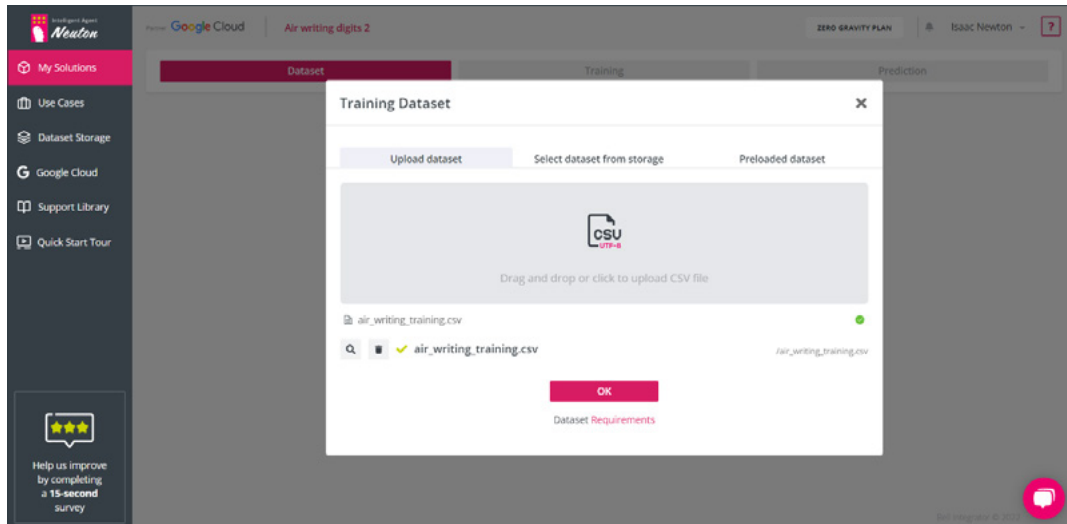
When the file is uploaded you can preview the selected dataset in the web interface using the **lens icon**.

If you have selected the wrong file by mistake you can click on the **trash** icon to select another dataset.

To go to the next steps, press "**OK**".

Select dataset from storage

Select one of the existing datasets by choosing the **"Select dataset from storage"** tab and navigate to a dataset that has been previously uploaded:

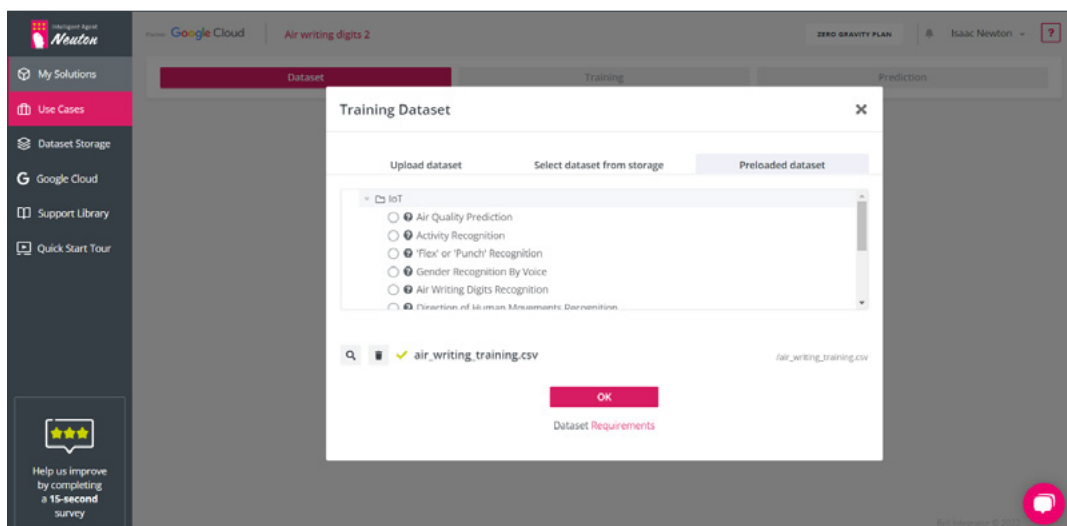


Picture 13 - Dataset uploaded

Preloaded Dataset

Select one of the preloaded datasets by choosing the **"Preloaded dataset"** option and navigate to the dataset that has been preloaded with Neutron.

To view information about the dataset, click on the corresponding **question mark** icon "?"



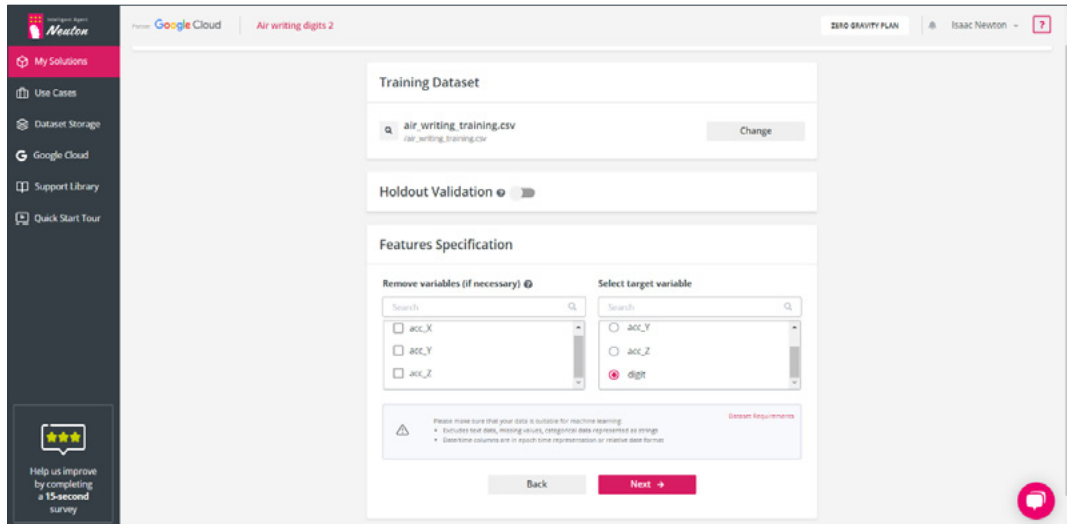
Picture 14 - Preloaded dataset

NOTE:

On the Dataset, Training and Prediction tabs for the preloaded datasets, all settings are preconfigured.

Specify Dataset Options

After the training dataset has been defined you should specify the target column. Also, at this stage, you can specify the dataset for holdout validation and drop some feature columns which you consider as irrelevant.



Picture 15 – Dataset options

To enable holdout validation and specify the dataset for it, turn on the switch button near **Holdout validation**. With the holdout dataset specified the training process will happen in the same way as without the holdout dataset but after model training completion metrics for the holdout dataset will be displayed instead of training metrics.

NOTE:

Holdout Validation dataset must be in the same format as the training dataset.

To exclude some features from the training dataset, mark the check box for the appropriate feature name in the **"Remove variables"** section. The model will not train on the excluded data.

NOTE:

If some features were dropped by using the **"Remove variables"** option in the web interface, the same features must be dropped manually by a user in the test data for inference on a device.

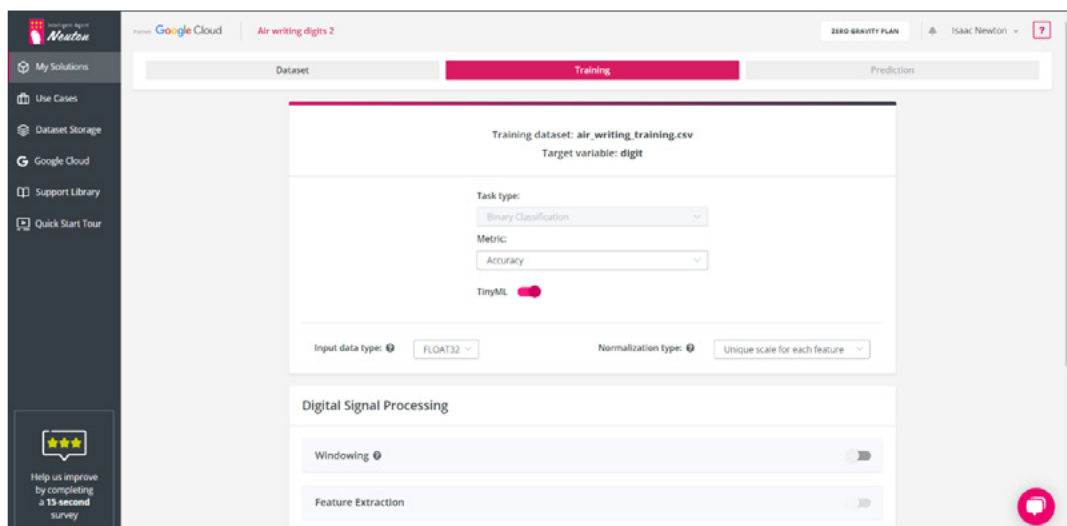
Click **"Next"** to proceed to the training stage.

2. Train your model

The model training process in Neutron contains the following stages:

- **Virtual Machine (VM) creation**
A VM is automatically provisioned to perform the following tasks: data preprocessing, feature engineering, model training and validation. This process is fully automated and does not require any user input or action.
- **Digital Signal Preprocessing (DSP)**
This option makes data suitable for creating models. DSP consists of windowing, feature extraction and feature selection.
- **Model training and validation.**
During this step, our proprietary neural network algorithm, Neutron, will automatically create a neural network architecture to achieve the best possible score on the validation data (measured by the metric selected by the user).
- **VM termination**
Upon completion of training, the virtual machine will be automatically deprovisioned.

You can control the training process via training parameters (settings). The following settings are automatically configured by Neutron however they can be changed manually.



Picture 16 – Training parameters

Training parameters:

Task type

Select the task type: Regression / Binary Classification / Multiclass Classification.

Task type parameter identifies the problem type which defines the basic model setting:

Neutron can automatically detect a task type based on the target variable values.

If Neuton automatically detected a task type as binary classification you can't switch it to another task type, but if Neuton detected a task type as multiclass classification you can switch it to regression and vice versa

- **Regression**

Regression is used for predicting the continuous numeric values of the target variable (e.g. price, temperature, age or any other real numeric value).

- **Binary Classification**

Binary Classification is used to predict one of the two possible outcomes or classes (e.g. 'yes' or 'no', 'black' or 'white', 0 or 1). If all of the values of your target variable are represented by only two unique values, this is a binary classification task type.

- **Multi Classification**

Multiclass Classification is used to predict one value of the limited number (greater than 2) of possible outcomes (e.g. 'red' or 'green' or 'yellow'; 'high' or 'medium' or 'low'; 1 or 2 or 3 or 4 or 5, etc.) If all of the values of your target variable are represented by a discrete (fixed) number of unique values/classes (>2), then this is a Multiclass classification task type.

Metric

Select the metric to measure model quality.

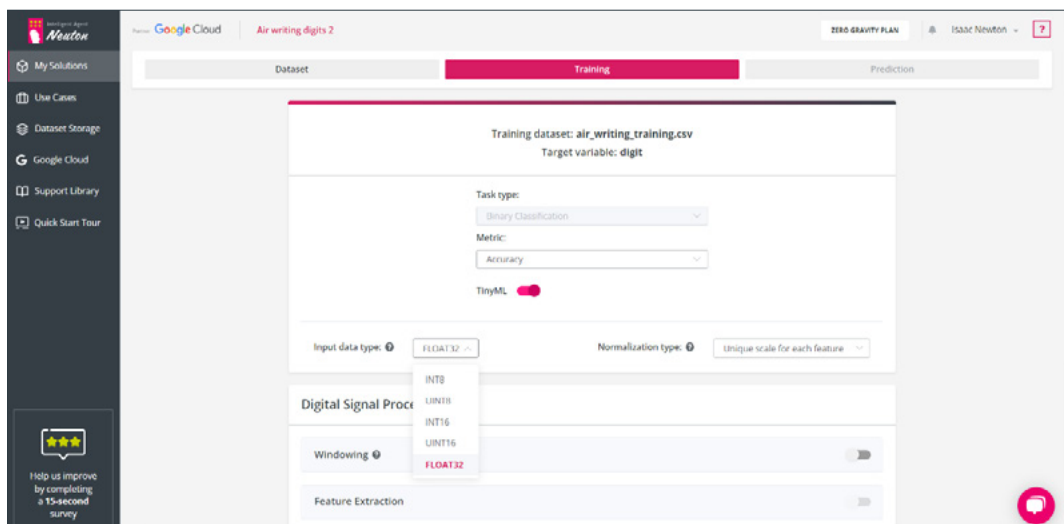
The target metric is a function calculating the error rate of the model predictions on the validation dataset. It represents the model quality (e.g. if for a classification task type, a model has correctly classified 7 samples out of 10, then the model Accuracy = 0.7). Detailed metrics descriptions are available in the **Metrics**.

During the training phase, the platform will measure the validation metric at each training iteration using a 10-fold cross-validation approach. Neuton has a built-in patented feature to prevent overfitting (overtraining) which stops training right before overfitting starts to occur. If the holdout validation dataset was specified final metrics will be calculated for it.

For Binary and Multiclass classification task types, the default value of the target metric is **Accuracy**, for the Regression task type the default target metric is **RMSE** (Root mean squared error, for more information please refer to "**Metrics definition**"). You can choose any applicable metric for the problem at hand.

NOTE:

The platform will recommend a "**Task type**" and "**Metric**" automatically by inspecting the target variable selected by a user, however, the user may change these parameters if necessary.



Picture 17 – Input Data Type

Input Data Type

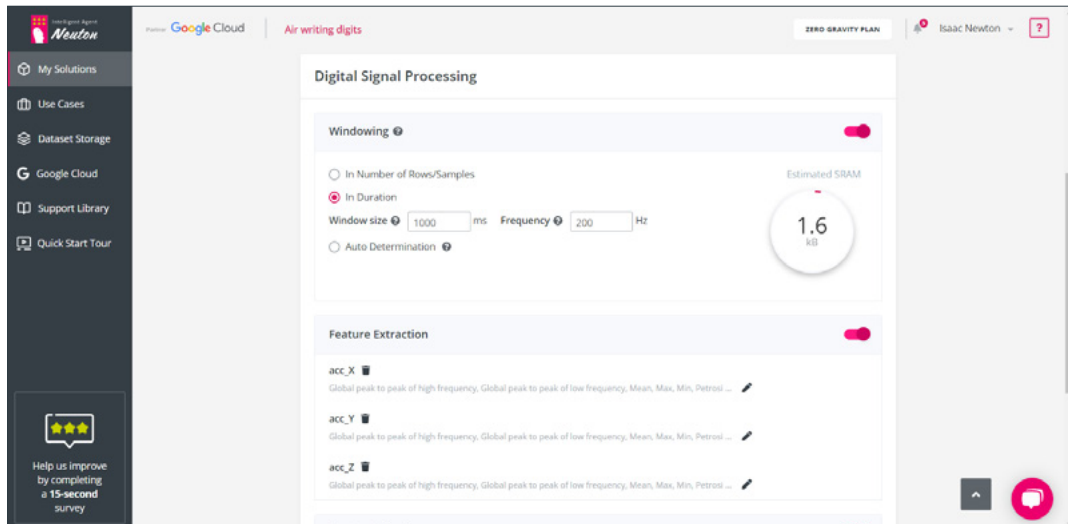
Select input data type according to the type of all features in the training dataset. For example, if you have a dataset with 3 features where 2 columns are in INT16 representation and the third column is in FLOAT32 representation you should select FLOAT32 data type for the training dataset. Input data type specification allows you to optimize preprocessing operation and reduce SRAM and Flash usage on the device during inference. The data of the same type must be used for prediction. Neutron supports the following input data types: INT8, UINT8, INT16, UINT16 and FLOAT32.

NOTE:

It is important to select the correct data type as incorrect selection of input data type may cause target metric degradation or model footprint increase.

Normalization type

This option allows you to select one of two scaling types for your dataset: “Unique scale for each feature” or “Unified scale for all features”. For model training normalization is used, unique scaling for each feature may increase model predictive power as well as model footprint, there as unified scaling may decrease model footprint. It is better to use “Unified scale for all features” when all values are in the same scale for all features.



Picture 18 – Digital Signal Processing options

Digital Signal Processing

Digital Signal Processing (DSP) option enables automatic preprocessing and feature extraction for data from gyroscopes, accelerometers, magnetometers, electromyography (EMG), etc. Neutron will automatically transform raw data and extract additional features to create precise models for signal classification. For solutions with the enabled **DSP** option, the **downloadable C Library** is the only available option on the prediction stage. For the DSP option, normalization type selection is not available, the system automatically selects one of the normalization types.

Windowing

Window size is the portion of data that will be used for processing the training dataset (the static window approach is applied on the platform). It should be universal for all events in the training dataset, even if the duration of events is different. Increasing of the window size will increase the model total footprint on the device but also may increase model accuracy. Using the window size the training dataset will be grouped by a selected number of samples. At this stage, the generation of lag features is happening, lag feature generation works for every feature with the selected window size. Lag feature generation may be turned on in the feature selection menu. For example, in the case of a 3-axis gyroscope (3 values per row in the training dataset) and “Window size”=50, 150 values will be stored in SRAM buffer.

You have 3 options for the windowing specification:

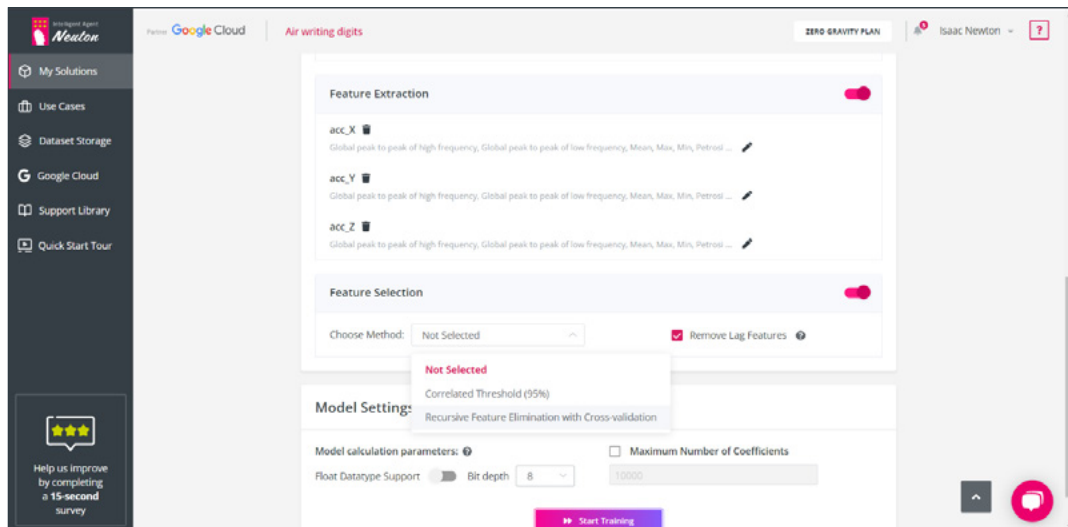
- **In Number of Rows/Samples**
Specify the count of rows to use for windowing
- **In Duration**
Specify window size in ms and frequency in Hz to define window size for the training dataset.
If you use sensors with different frequencies, then you must first bring all signals to a single frequency using downsampling or upsampling

- **Auto Determination**

Using Fast Fourier Transform this option automatically selects the window size based on the best model quality (this option is available only for the classification task type). For finer tuning of SRAM usage, it is recommended that you manually enter the window size and frequency of data.

Estimated SRAM Usage

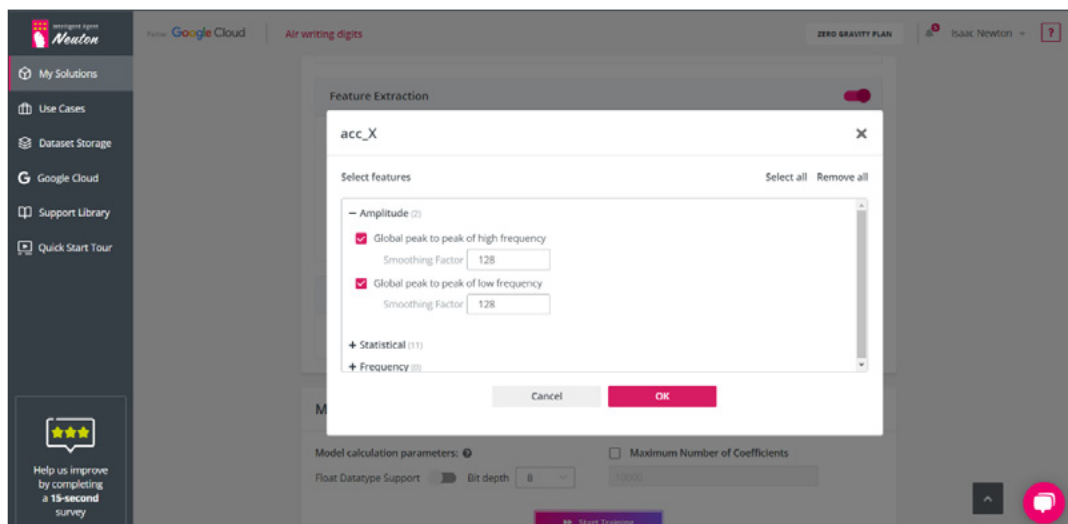
Based on the selected windowing size, the Neutron platform shows estimated SRAM usage in kB.



Picture 19 – Feature Extraction, Feature Selection and Model Settings

Feature extraction

When the window size is specified, the platform automatically extracts the following features for each column in the training dataset. The same feature extraction will be executed during inference on the device. You can manage calculated features for each original variable in the training dataset by marking or unmarking the appropriate checkbox.



Picture 20 – Extracted features

The list of supported features is provided below:

Amplitude

- **Global peak to peak of high frequency**
Global peak to peak of high frequency is a calculating of the high-frequency signal by subtracting the moving average filter output from the original signal (feature). You can specify the Smoothing Factor which is the amount of attenuation for frequencies over the cutoff frequency. The smoothing factor value should be equal to the power of 2 but not greater than the window size.
- **Global peak to peak of low frequency**
Global peak to peak of low frequency is a calculating of the low-frequency signal by applying a moving average filter with a smoothing factor. The smoothing factor value should be equal to the power of 2 but not greater than the window size.

Statistical

- **Mean**
The Mean function calculates the arithmetic mean of the column (feature).
- **Max**
The Max function calculates the maximum of the column (feature).
- **Min**
The Min function calculates the minimum of the column (feature).
- **Petrosian fractal dimension**
This function converts the data to a binary sequence and estimates the fractal dimension from the time series.
- **Variance**
The Variance function calculates the minimum of the column (feature).
- **Root mean square**
The root mean square is the root of the arithmetic mean of the squares of a set of numbers.
- **Skewness**
The skewness function measures the asymmetry of the distribution of a variable.
- **Kurtosis**
Kurtosis is a measure of the combined weight of a distribution's tails relative to the center of the distribution.
- **Mean crossings**
This function calculates the number of times the selected column crosses the mean.
- **Negative mean crossings**
Computes the number of times the selected input crosses the mean with a negative slope.

- **Positive mean crossings**

Computes the number of times the selected input crosses the mean with a positive slope.

Frequency

- **FFT first peak power**
- **FFT first peak frequency**
- **FFT second peak power**
- **FFT second peak frequency**
- **FFT third peak power**
- **FFT third peak frequency**

These functions use fast Fourier transform to calculate the frequency of peaks and their power. If Frequency features are selected the window size should be equal to the power of 2.

Feature selection

- **Remove Lag Features**
You should uncheck the check box to use lag features for the model training otherwise the lag features will not be used for model training.
- **Remove Correlated Features**
Check the box to remove correlated features before model training, the correlation threshold is 95%.
- **Recursive Feature Elimination with Cross-validation**
With this option enabled Recursive feature elimination will be executed by diverse machine learning algorithms with cross-validation.

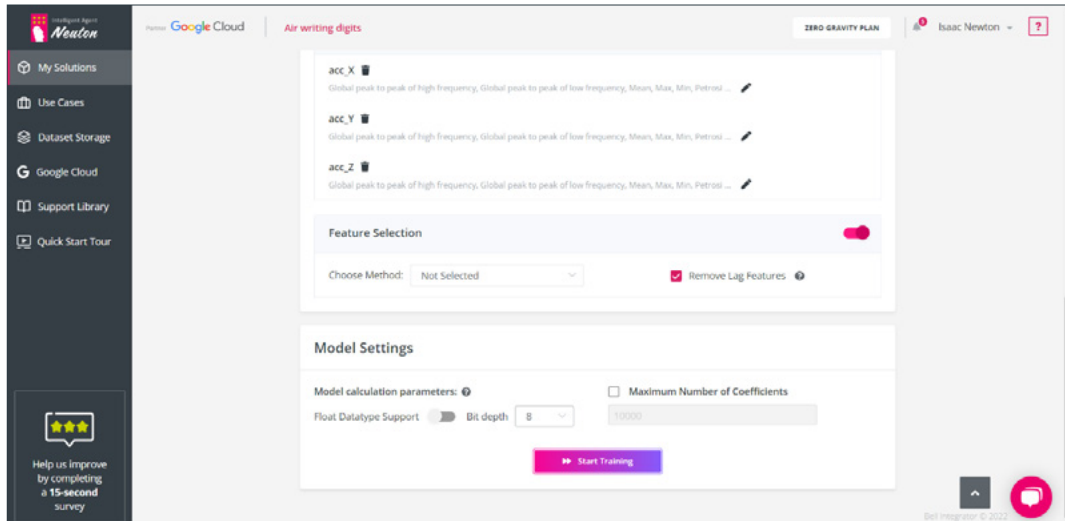
Model Settings

- **Model calculation parameters**
Select the number of bits (8, 16 or 32) that your model should support. With Float Datatype Support enabled you can build models that support operations with the float data type (**Float Datatype Support** is always available for 32-bits calculations). By default, the same
- **Maximum Number of Coefficients**
The number of coefficients represents the size of the model and the required number of operations necessary for predicting with the model. Neuton models are very small by default, however you may still limit the number of coefficients if necessary. By default, the number of coefficients is unlimited.

After the training parameters are defined click "**Start training**" to start the training process.

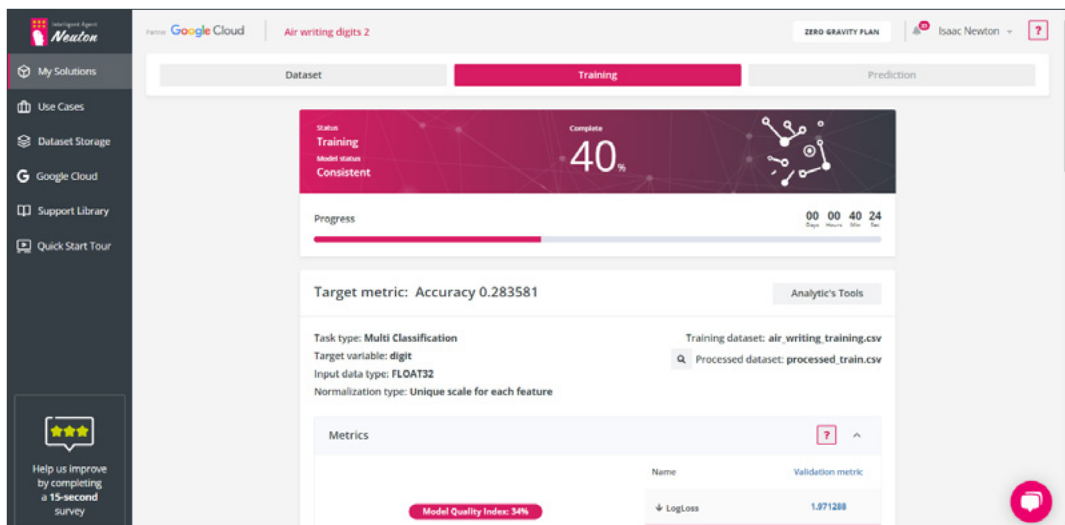
Start and control the Model training process

When you have specified the training parameters you can click the **"Start Training"** button (see **picture below**).



Picture 21 – Start Training

After the training is initiated you will see a **Training** process progress bar:



Picture 22 – Training Progress

By default the Training progress window shows the following parameters:

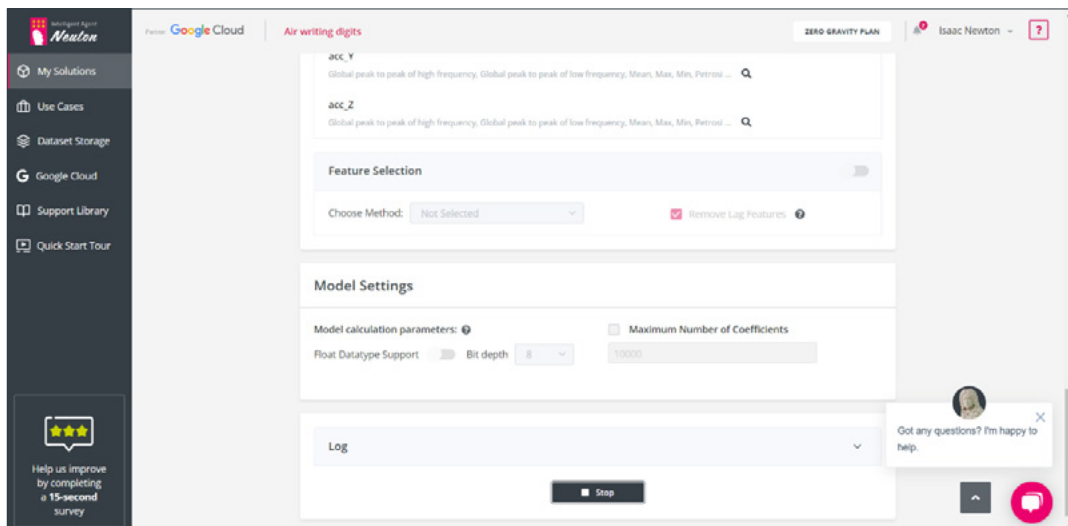
- Model consistency. Model status **"Not consistent"** or **"Consistent"** is displayed during the training process
- **Progress bar** and **percentage**
- Selected **Task type**
- **Training dataset** name
- **Target variable** name

- Selected **Input Data Type**
- **Target metric**
- **Analytics Tools**
- **Metrics**
- **Log**

NOTE:

After the Digital Signal Processing stage (if enabled), and model training are complete, you can review the processed training dataset for your solution by clicking the "**lens**" icon near "**Processed dataset**".

When a model is consistent you have the option to stop the training ("**Stop**" training button). In this case, the best model will be saved and available for prediction in a few minutes.



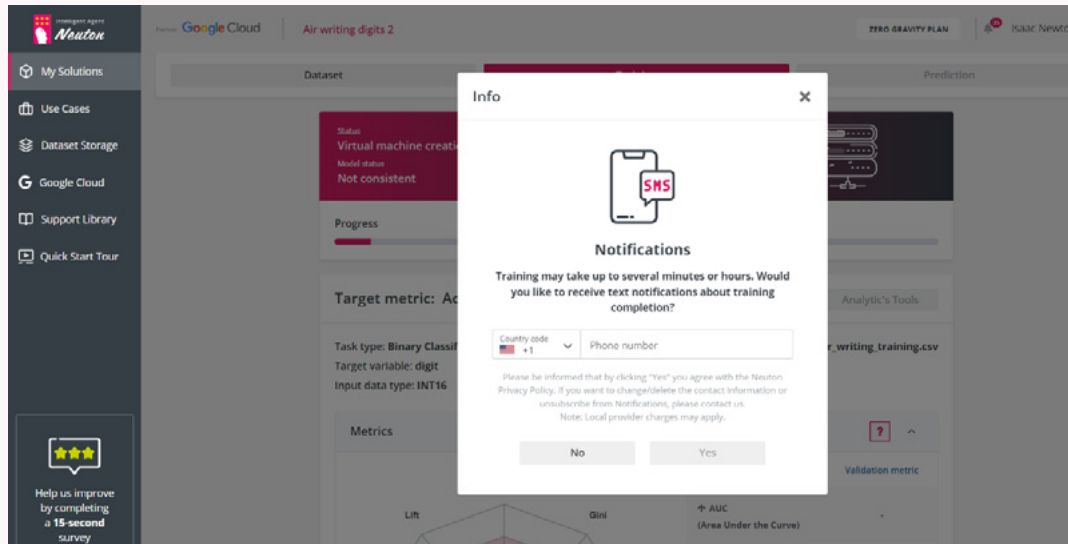
Picture 23 – Stop Training button

NOTE:

If the user has initiated the "**Stop**" training option the training infrastructure will be de-provisioned and the user will not incur any further infrastructure costs for training. The most optimal model will be saved. After stopping you can not restart or resume model training for the same solution.

NOTE:

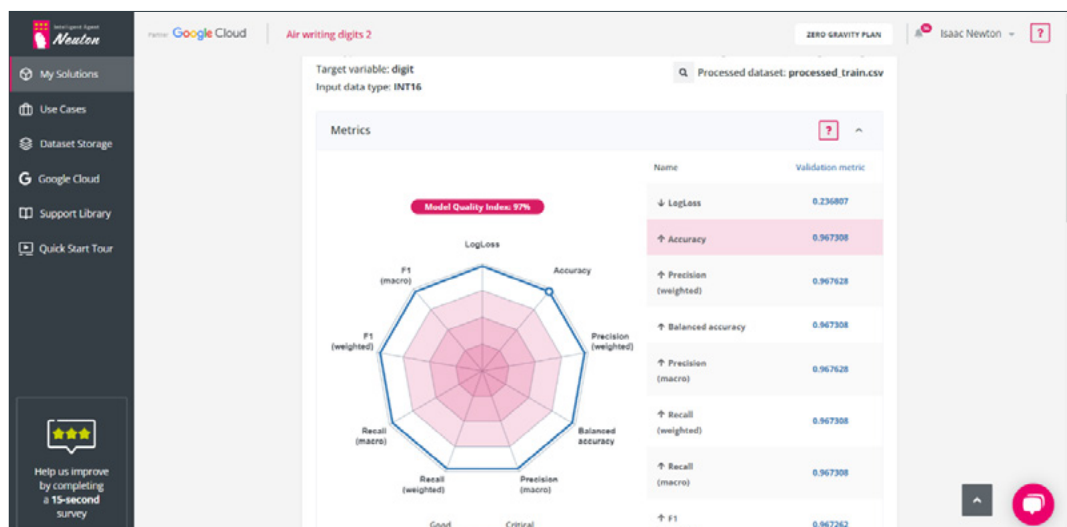
Sometimes training may take up to several minutes or several hours depending on dataset size and data structure. For your convenience, Neuton can inform you about training completion with text notifications. You can provide the best phone number for receiving updates in the pop-up window when training is initiated. Please note that local provider charges may apply. Also, by clicking "Yes", you agree with the Neuton Privacy Policy.



Picture 24 - Text Notification

Metrics

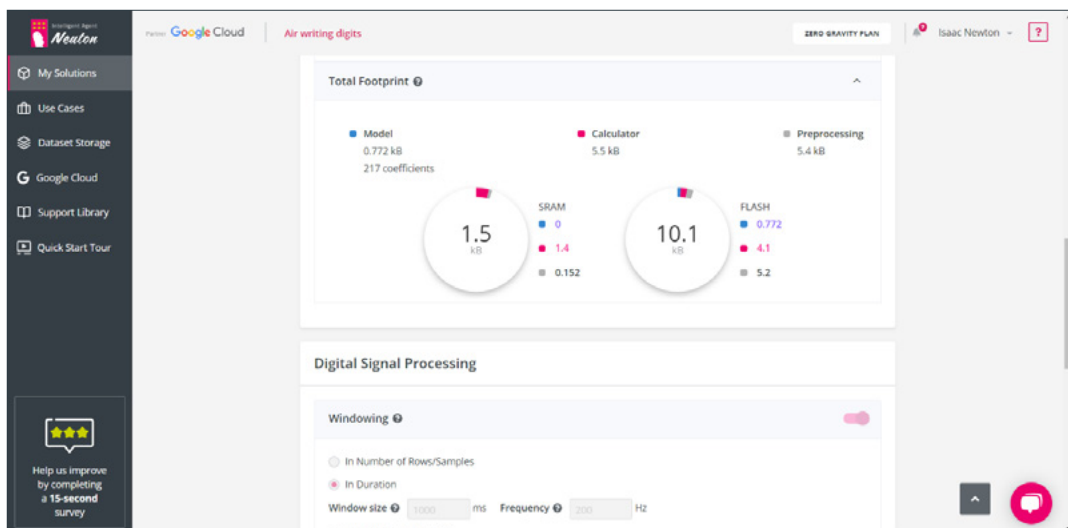
Despite the **Target metric** (selected by user) the platform additionally calculates various useful metrics applicable for each task type (for user convenience) which you can observe. For the metric description, please refer to the "**Metrics definition**" section.



Picture 25 – Metrics, Model Quality index and Diagram

In Metrics you can also find the following information:

- **Model Quality index**
This Index determines the quality of the model based on the metric indicator values.
- **Model Quality Diagram**
For any task type Neutron builds **Model Quality Diagram**, for more information please refer to the **Explainability Office** section.
- **Coefficients**
This value represents the current number of neural network coefficients
- **Total Footprint with Estimated SRAM and FLASH usage**
Here you can find information on the estimated usage of SRAM and FLASH memory by the model, calculator and preprocessing. Estimated values are provided for Arduino M0 Pro (Cortex M0). These values are available only after model training completion.



Picture 26 – Total Footprint

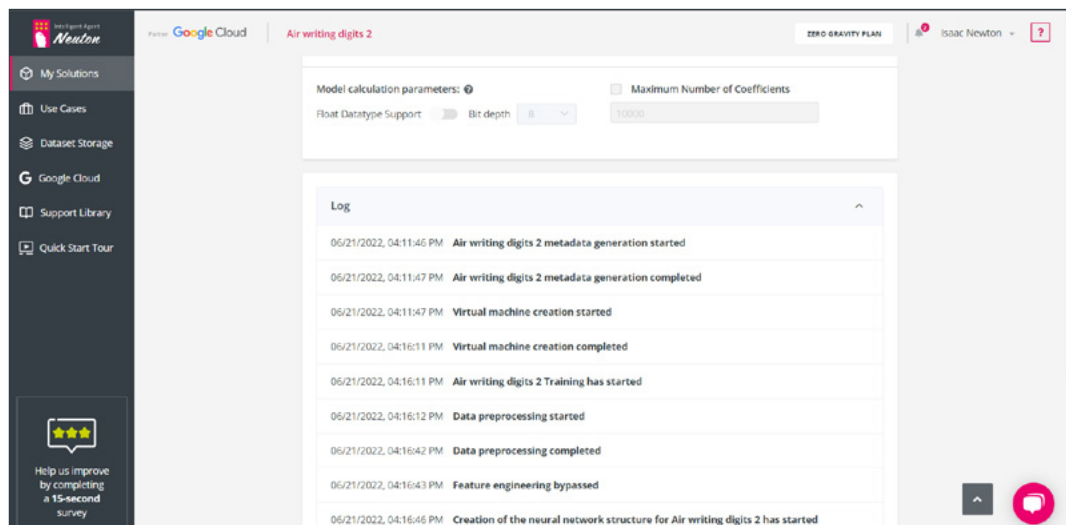
Analytics Tools

The "**Analytic's Tools**" button allows users to open the following components in Neutron web Interface:

- **Analytic's Tools. Exploratory Data Analysis (EDA).**
EDA is available after DSP stage completion.
- **Analytic's Tools. Feature importance matrix (FIM).**
FIM is available on the Prediction stage.

Log

During the training process, you can monitor training steps in the Log:



Picture 27 – Log

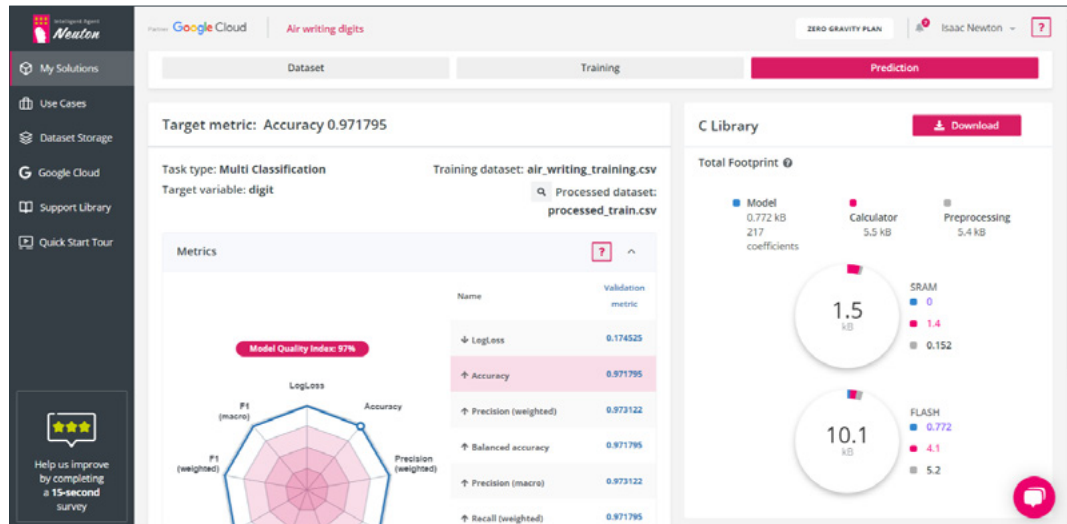
The following steps are being logged:

- **Solution metadata generation started**
Platform performs initial data analysis.
- **Solution metadata generation completed**
Platform finished initial data analysis.
- **Virtual machine creation started**
Provisioning of infrastructure for training.
- **Virtual machine creation completed**
Infrastructure has been successfully provisioned.
- **Training started**
Training pipeline is initiated..
- **Data preprocessing started**
Digital Signal Processing is the process that is started.
- **Data preprocessing completed**
Digital Signal Processing has been completed successfully.
- **Data preprocessing bypassed**
Data preprocessing option was unselected by a user and is not performed.
- **Creation of the neural network structure started**
Data is fully prepared and Neutron neural network is being built.
- **Creation of the neural network structure completed**
Model building is completed and can be used for predictions/download/ deployment/etc.
- **Training completed**
Platform finished model training.
- **Training interrupted**
Max Number of Coefficients was reached.

3. Make Predictions

Using the AI Model for Prediction

After the model training is completed you will be redirected to the "**Prediction**" tab.



Picture 28 – Prediction Tab

The "**Prediction**" tab includes:

- **Info area**
Info represents an overview of your trained Solution.
- **C Library**
C Library allows users to download the resulting model and code for inference on the device.
- **Web prediction for disabled DSP**
Web prediction allows users to start predictions on new data with the Neutron web service. Web prediction is available only for solutions with the turned-off DSP option
- **REST API Access for disabled DSP**
REST API Access allows third-party applications access to the model. REST API is available only for solutions with turned off DSP option.

Info Area

In the Info area you can see the following information:

- **Target metric** name and its value
- Selected **Task type**
- **Target variable** name
- **Training dataset** name
- List of **Metrics** calculated during training including **Model Quality Diagram** and **Model Quality Index** (For more information about supported metrics for each task type please refer to "**Metrics Definition**", and to learn more about

the **Model Quality Diagram** and **Model Quality Index** please refer to the **Explainability Office** section.)

- **Analytic's Tools** (See **Explainability Office** section)
- **Total Footprint**
- **Model --to --data relevance indicator**

C Library

Neuton, our unique neural network framework, natively creates incredibly compact and accurate models that can easily be deployed into your firmware project using an automatically generated archive with a C library.

The library is written in accordance with the C99 standard, so it is universal and does not have strict requirements for the hardware. The ability to use the library depends mainly on the amount of memory available for its operation.

The archive contains the following files and folders:

- **neuton.h** - header file of library
- **neuton.c** - library source code
- **model/model.h** - model header file
- **StatFunctions.h / StatFunctions.c** - statistical functions for preprocessing

NOTE:

We don't recommend that you modify any files in the archive. Unsupervised changing of files may cause errors in model inference.

- **Converted_models**

High-level integration steps include:

1. Copying all files from the archive to the project and including the header file of the library.
2. Creating a float array with model inputs and passing it to ``neuton_model_set_inputs`` function.
3. Calling ``neuton_model_run_inference`` and processing the results.

In the instruction below the detailed information about how to integrate Neuton into your firmware project is provided.

How to integrate Neuton into your firmware project

Include header file

Copy all files from this archive to your project and include the header file:

```
#include "neuton.h"
```

The library contains functions to get model information such as:

- task type (regression, classification, etc.);
- neurons and weights count;
- window buffer size;
- input and output features count;
- model size and RAM usage;
- float support flag;
- quantization level.

Main functions are:

- ``neuton_model_set_inputs`` - to set input values;
- ``neuton_model_run_inference`` - to make predictions.

Set input values

Make a float array with the model inputs. Input count and order should be the same as in the training dataset.

```
float inputs[] = {  
    feature_0,  
    feature_1,  
    // ...  
    feature_N  
};
```

If the digital signal processing option was selected on the platform, you should call ``neuton_model_set_inputs`` multiple times for each sample to fill the internal window buffer. The function will return ``0`` when the buffer is full, this indicates that the model is ready for prediction.

Make predictions

When the buffer is ready, you should call ``neuton_model_run_inference`` with two arguments:

- pointer to ``index`` of predicted class;
- pointer to neural net ``outputs`` (dimension of array can be read using ``neuton_model_outputs_count`` function).

For regression task output value will be stored at `outputs[0]`.

For classifications task `index` will contain a class index with maximal probability, `outputs` will contain probabilities of each class. Thus, you can get predicted class probability at `outputs[index]`.

Function will return `0` on successful prediction.

```
if (neuton_model_set_inputs(inputs) == 0)
{
    uint16_t index;
    float* outputs;

    if (neuton_model_run_inference(&index, &outputs) == 0)
    {
        // code for handling prediction result
    }
}
```

The same instruction you can find in the downloaded archive in **README.md** file.

Integration with Tensorflow, ONNX

For processed data (for solutions with turned off DSP option) and 32FLOAT input data type created models are also available in TensorFlow and ONNX formats which can be built in any pipeline. You can find a model in these formats in the `converted_models` folder.

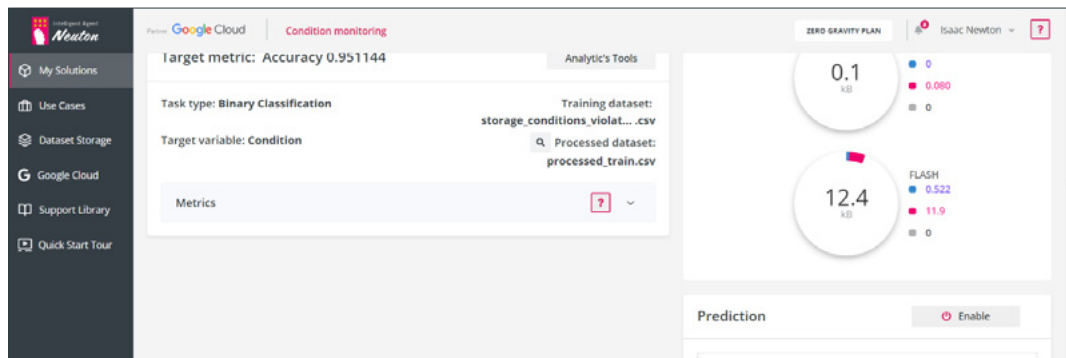
NOTE:

- The input data for predictions must be identical in format to the data used for the training, including the order of columns. The target variable should be excluded. If some columns were dropped using the platform web interface the same columns should be dropped in new data for prediction.
- The C Library returns predicted classes in encoded representation, to convert them to original representation please refer to README.md file where you can find all necessary information.
- Models in Tensorflow and ONNX formats are available only for the 32-bit version of the models.
- In model.h file you can find features which are marked as “unused by model” that means that these features are not used by the model during inference, but you can not drop these features from data used for inference. To optimize model performance, you can train the new model on the new training dataset with dropped features which were marked as “unused by model”. In these case, you can drop the same features in the data used for prediction.

Web Prediction

To predict using new data you can use a web interface. The service can be enabled by clicking **"Enable"**. This option is available only for solutions with disabled Digital Signal Processing.

Web prediction requires cloud infrastructure to process new data, calculate predictions and deploy models. Enabling the prediction service will result in some infrastructure charges. Confirmation of the associated charges is required.



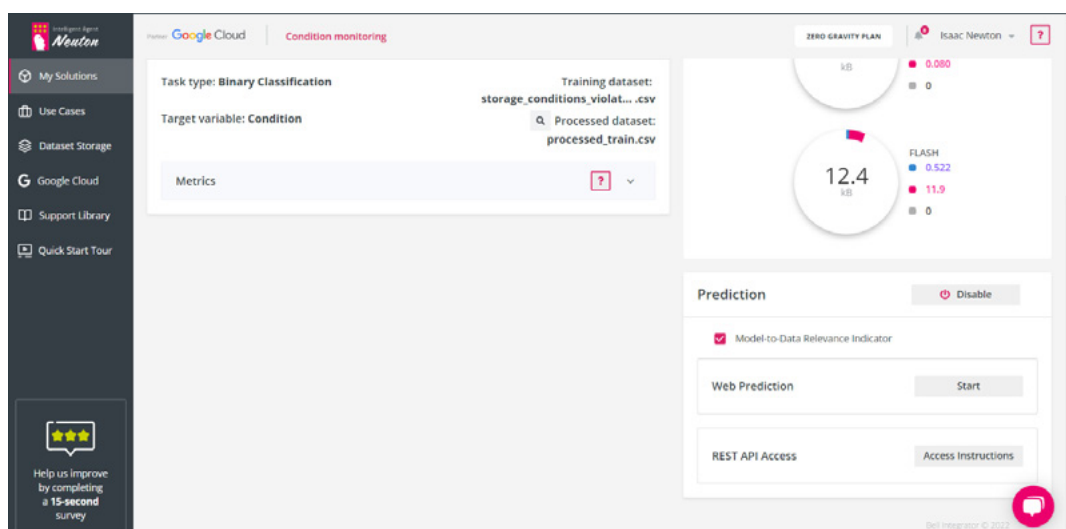
Picture 29 - Enable prediction

NOTE:

At the time these infrastructure charges are accepted, monthly bills will be generated and delivered in accordance with our pricing policy.

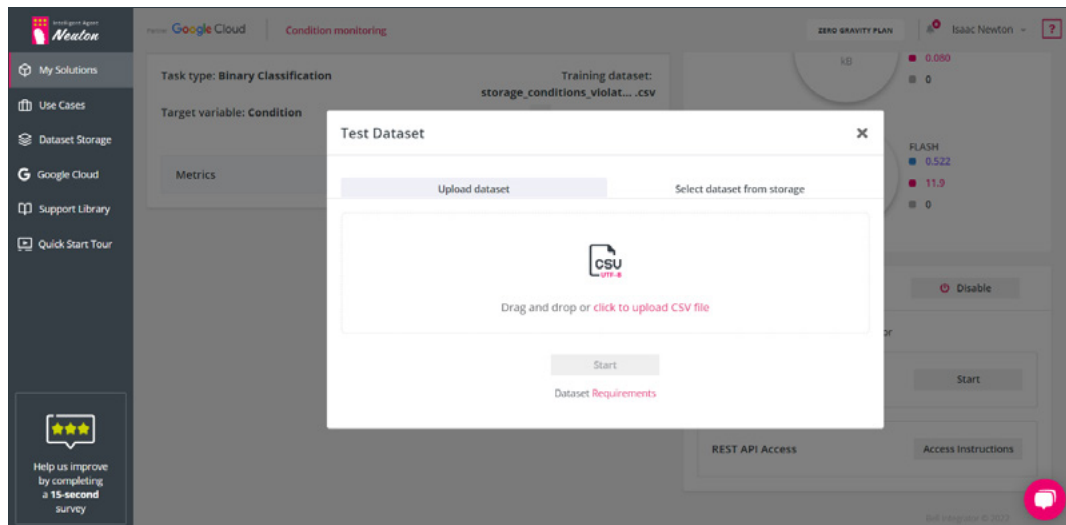
Click **"OK"** to start the virtual machine creation for prediction.

After the service is enabled, the **"Web Prediction"** and **"REST API Access"** buttons will become active:



Picture 30 - Prediction enabled

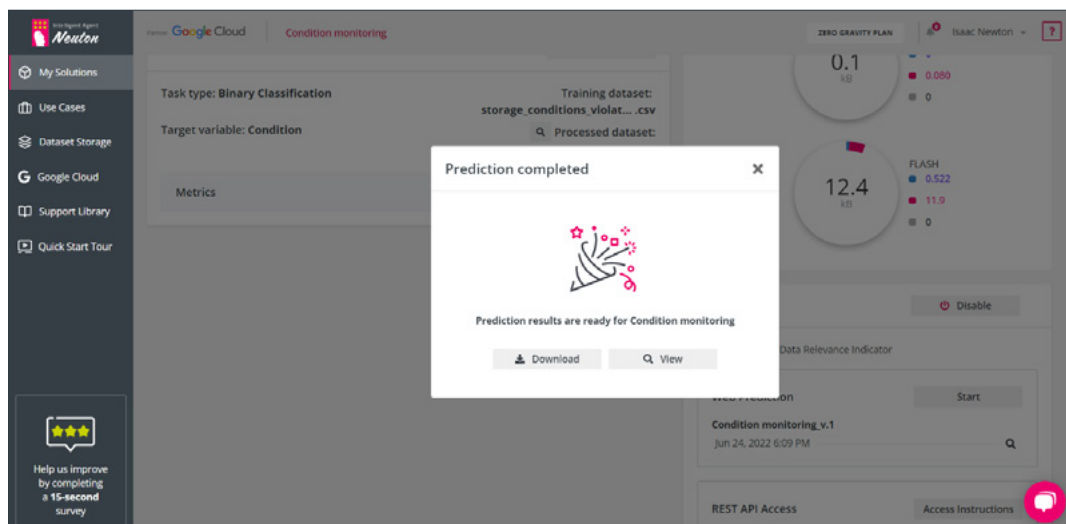
Click **"Start"** next to **"Web Prediction"** to upload new data.



Picture 31 - Upload dataset for Predictions

The dataset uploading/selection process is similar to the process described in the **"1. Select Data for Training"** section. The test dataset (new data) must have the same structure as the training dataset, except there should be no target variable. For preloaded training datasets, the test dataset is preselected.

After the dataset is uploaded / selected, click **"Start"** to start the prediction process. When the prediction is completed, the system will display the following message:



Picture 32 - Prediction results are ready

Users may choose to **"Download"** the predictions to a local hard drive as a csv file or **"View"** the prediction results in the browser window. If **"View"** is selected, the predictions will appear concatenated to the existing data (uploaded for prediction):

Nr	Condition	Probability of 0	Probability of 1	Model to Data Relevance Indicator	Humid	Temp
1	0	0.999965	0.000035	1	26.2573	22.248
2	0	0.853557	0.146443	1	36.548	22.1493
3	0	1.000000	0.000000	1	29.89	34.23
4	1	0.277170	0.722830	1	46.6987	20.9173
5	0	1.000000	0.000000	1	30.11	34.16
6	1	0.000070	0.999930	1	12.35	49.26
7	0	0.912097	0.087903	1	41.3827	23.248
8	0	1.000000	0.000000	1	25.58	34.53
9	1	0.002214	0.997786	1	57.6107	18.7313
10	0	0.999960	0.000040	1	29.7907	23.2933

Picture 33 - View prediction results

You can see the predicted values in the first column. Click on the timestamp under the **"Start"** button to see the prediction results you have made previously. On this step also available Model Interpreter (See Explainability Office Section)

NOTE:

- For binary and multiclass classification task types, you will find additional information in prediction results - the **probabilities** of the predicted classes.
- For the regression task type, in prediction results you can find the **Confidence Interval** and **Confidence Probability**. Confidence Interval with calculated probability, shows the possible prediction spread for each predicted value. For example, you have predicted a house value to be 200000 USD. With a 95% probability the possible range for this prediction may be: +/- 5000 USD.
- For any task type Neutron calculates the **Model-to-Data Relevance Indicator**. Taking into account feature impact on the model prediction, the Model-to-Data Relevance Indicator calculates the statistical similarity between the data uploaded for predictions and the data used for model training. The measure (100-M2D) shows the possible degradation level for the target metric value.

A low value for M2D may indicate a significant change in the model input data (data drift) that can lead to model performance degradation (model decay).

For example, if the Model-to-Data Relevance Indicator on the current dataset (uploaded for predictions) equals 95%, then the user can reasonably expect the model quality to decrease by as much as 5% from the validation metric.

100% = no change in model input data, no model performance degradation.

1% = a significant change in model input data, that leads to significant model performance degradation.

Model-to-Data Relevance Indicator is calculated for:

- every row sent for predictions
- dataset sent for predictions
- all data sent for predictions aggregated over time (Historical Model-to-Data Relevance Indicator).

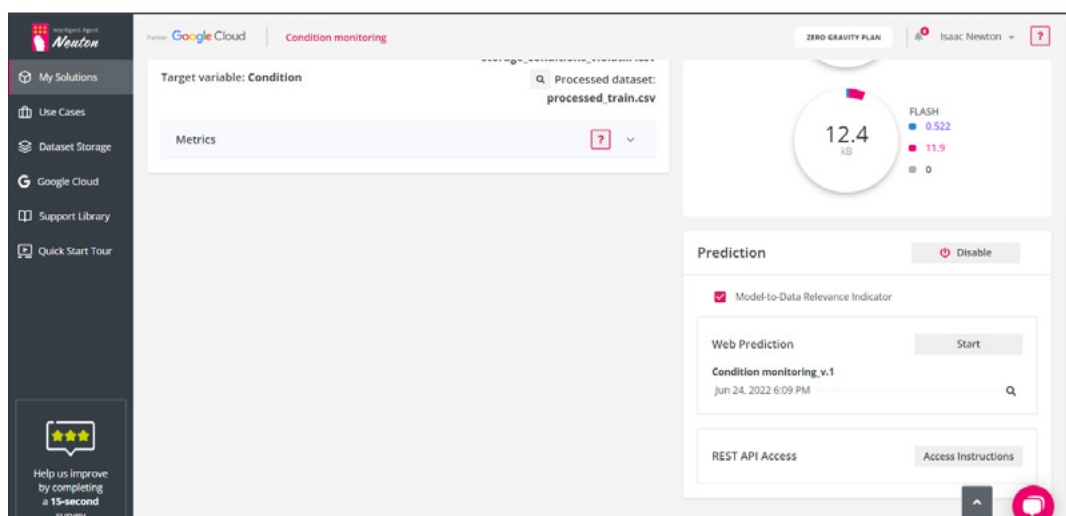
The structure of prediction results is the same for any kind of prediction (WEB, REST API or using Downloadable Solution).

REST API access

When **Prediction** is enabled, **REST API Access** to the model is also enabled automatically. With **REST API Access** users can send new data for predictions to the hosted model from anywhere on the web, for example, you have developed your own application which you would like to augment with Neuton AI capabilities by sending prediction requests to the Neuton platform. Click **Access Instructions** (see **picture below**) on the **Prediction** tab to copy request codes.

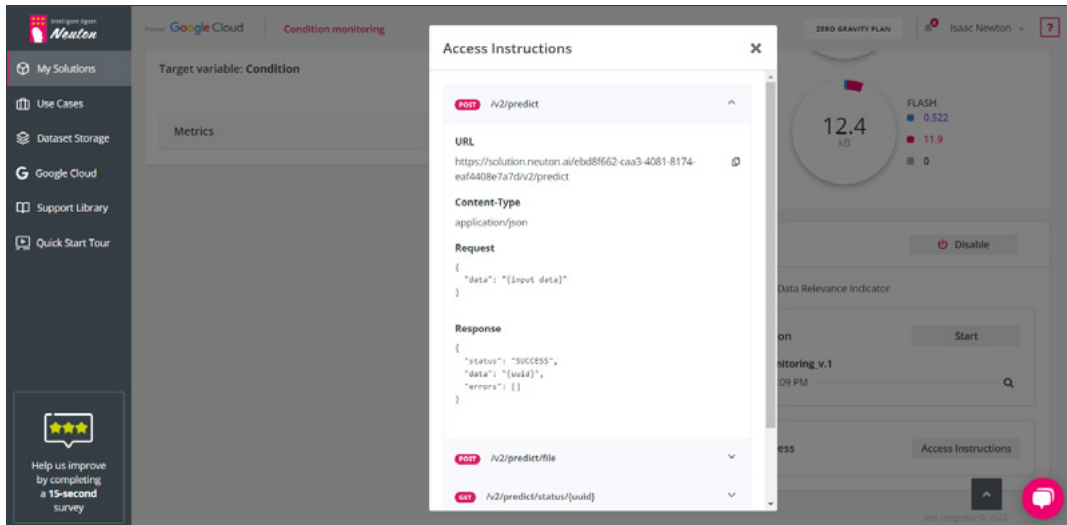
In the pop-up window, you will find following instructions:

- **/v2/predicton**
send a request for prediction of a single line;
- **/v2/predict/file**
send a request for prediction of a whole file;
- **/v2/predict/status/{uuid}**
possible returned statuses;
- **/v2/predict/result/{uuid}**
get request example for prediction results.



Picture 34 – REST API Access

You can expand instructions and view **URL**, **Content-Type**, **Request** and **Response** codes.



Picture 35 – REST API access instructions expanded

Additionally, you can download the code samples by clicking **"Samples"**. You can find code samples for predicting on a whole file for multiple programming languages including Scala, C#, Java and Python to access the deployed model via the REST API service.

To predict on the new data, you should specify the following parameters in code samples before running it:

- **url**
the url from the Access Instructions, which is equal to the url address of the hosted model before the 4th slash, but not including it (eg, url='https://solution.neutron.ai/699de50c-11df-2742-85e0-5dad58e3197b').
- **file_path**
the path to test the dataset or new data (eg, file_path=' test.csv').

NOTE:

If the **Web Predictions** and/or **REST API Access** to the model is no longer required users should **Disable** prediction service to optimize the use of infrastructure.

Disabling Predictions

If a user does not need to use the model for further predictions, then the model may be 'undeployed' by clicking **"Disable"**. The prediction VM will be deprovisioned. Note that after disabling the model will be stored on the platform and user can enable it for predictions at any time.

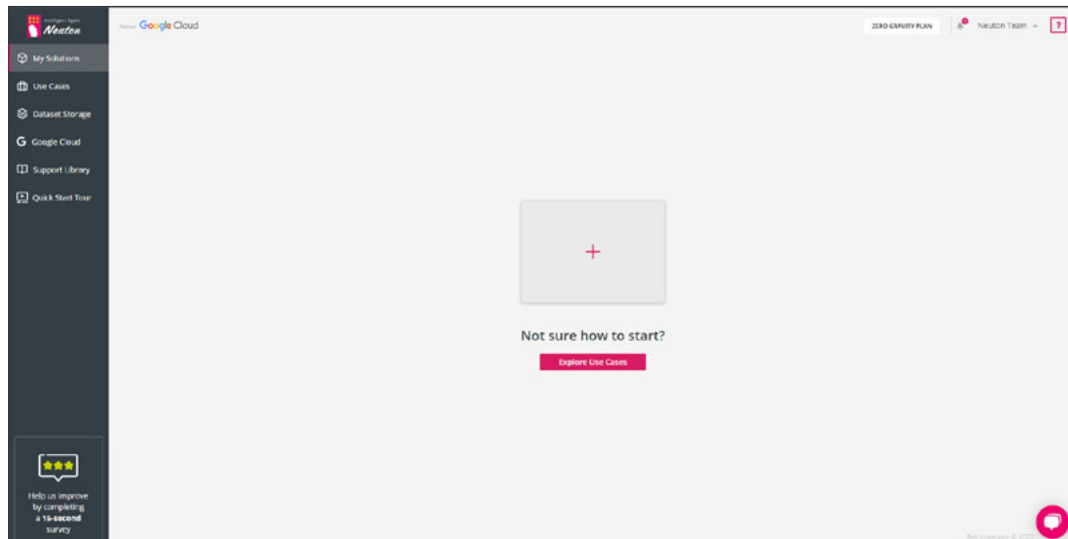


Picture 36 – Disable predictions

AUDIO FILE PROCESSING

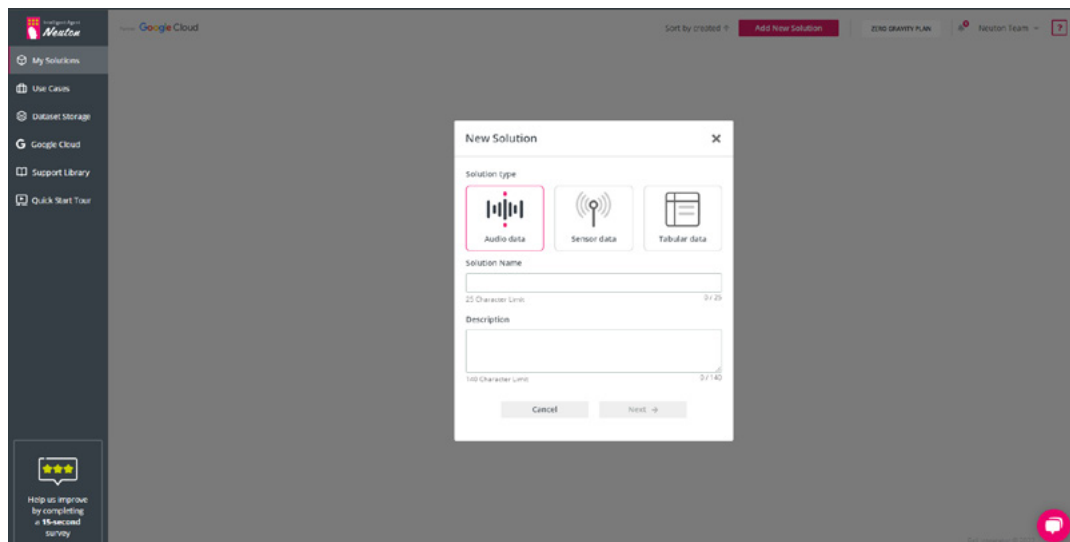
1. Preparing data for training

On the starting page **My Solutions**, click the **Add New Solution** button.



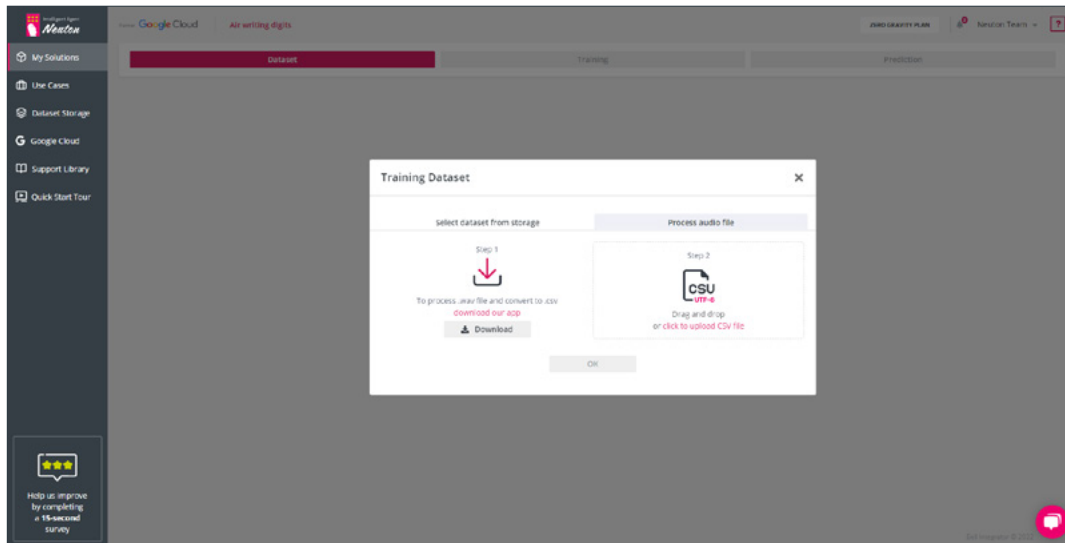
Picture 37 – My Solutions. Default View

In the pop-up window, choose the **Input Data Type** (Audio Data), specify a **Solution Name**, and click **Next**.



Picture 38 – New Solution. Pop-up Window

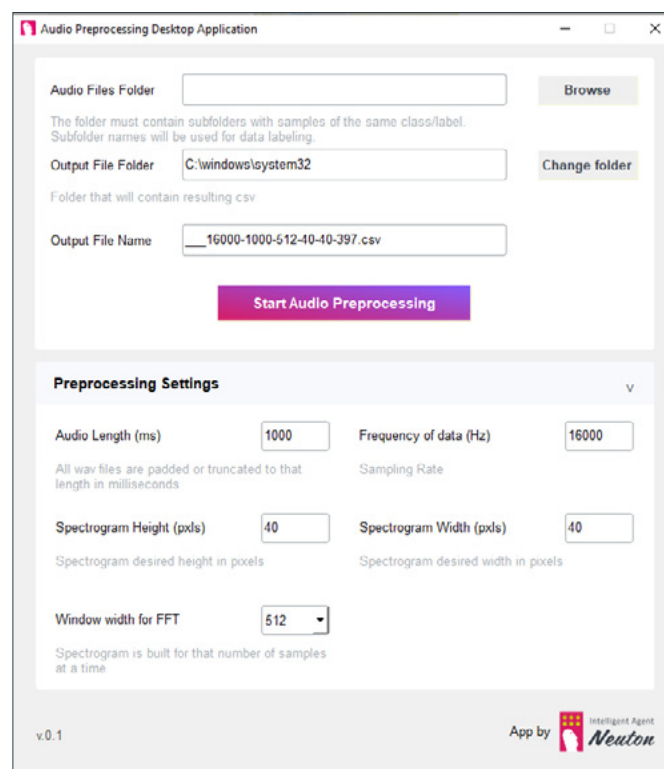
If you have a ready-made dataset processed with our application, you can select it from a storage.



Picture 39 – Select Dataset for Training

Otherwise, download our **Audio Preprocessing Desktop Application** to process audio files in wav or wave formats and create your dataset.

Neutron audio preprocessing app requires no installation. When downloading is complete, you can run the app to specify file path and preprocessing parameters.



Picture 40 – Audio Preprocessing Desktop Application

The following settings are available in the application:

- **Audio files folder**
This is a path to your raw audio files in wav or wave formats. The folder must contain subfolders with samples of the same class (label). Subfolder name will be used for class labeling.
- **Output file folder**
The folder for resulting processed CSV file.
- **Output file name**
The file name is generated automatically. It is strongly recommended to leave file name unchanged as file name is used for parsing of audio processing parameters during model training.
- **Spectrogram Height (pxls)**
Specify desired spectrogram height in pixels.
- **Spectrogram Width (pxls)**
Specify desired spectrogram width in pixels.
- **Frequency of data (Hz)**
Specify sampling rate in Hz.
- **Audio length (ms)**
Specify the duration of audio files in ms.
- **Window width for FFT**
Specify the number of samples for which spectrogram will be built using fast Fourier transform.

Click the **Start Audio Preprocessing** button.

Your CSV file will be saved to the output file folder.

Drag and drop the generated dataset.

Click on the **lens icon** to see the dataset details.

To calculate metrics on the new portion of data that won't be used in model training, upload a holdout validation dataset. You have to convert your wav files to the validation dataset with the same settings as in the training dataset using our application

Then click **Next** and you will be redirected to the **Training tab**.

2. Model training

On the Training tab, set the following parameters:

Task type

Select the task type: Binary Classification / Multi Classification / Regression.

Task type parameter identifies the problem type which defines the basic model setting.

Neuton can automatically detect a task type based on the target.

Metric

Select the metric to measure model quality.

For Binary and Multi classification task types, the default value of the target metric is Accuracy.

NOTE:

The platform will recommend a **Task type** and **Metric** automatically by inspecting the target variable selected by a user, however, the user may change these parameters if necessary.

Digital Signal Processing

The Digital Signal Processing block displays all the settings you have selected in our app. They are not editable. If you see that you need to change some settings, you will need to return to the app, reload wav. files, select the correct settings, and upload the resulting file into the platform.

Model Settings

- **Model calculation parameters**

Select the number of bits (8, 16 or 32) that your model should support. With Float Datatype Support enabled, you can build models that support operations with floats.

- **Maximum Number of Coefficients**

Neuton models are very small by default, but you may still limit the number of coefficients if necessary. By default, the number of coefficients is unlimited.

The screenshot shows the 'Training' settings page. At the top, it displays the training dataset as 'train_subset.wav' and the target variable as 'label'. Below this, there are dropdown menus for 'Task type' (set to 'Binary Classification') and 'Metric' (set to 'Accuracy'). The 'Digital Signal Processing' section, marked with an 'Audio' icon, shows parameters: Audio length/Window size (ms): 1000, Spectrogram height (pxls): 40, Window width for FFT: 512, Frequency of data (Hz): 16000, and Spectrogram width (pxls): 40. The 'Model Settings' section includes 'Model calculation parameters' with a help icon, 'Float Datatype Support' (enabled with a pink toggle), 'Bit depth' (set to 32), and 'Maximum Number of Coefficients' (set to 100000). The 'Footprint' section shows 'MCU Type' as 'Arduino M0 Pro (ARM Cortex-M0)'. A 'Start Training' button is at the bottom right.

Picture 41 – Training Settings

After the training parameters are defined click **“Start training”** to start the training process.

Model training may take from several minutes to several hours depending on the dataset size. For your convenience, you may leave your phone number and be notified about the end of training via text messages.

3. Model downloading

Once the training is complete, you can download the ready-to-use C library. The archive contains all the necessary files and instructions for model embedding and running inference on the device.

EXPLAINABILITY OFFICE

The Explainability Office is a single pane of view for Neuton's Explainability features:

- **Exploratory Data Analysis (EDA)**
- **Model Quality Index**
- **Model Quality Diagram**
- **Feature Importance Matrix (FIM)**
- **Model Interpreter**

Exploratory Data Analysis (EDA)

The EDA is the first explainability feature inside the Neuton model life cycle. This tool automates initial data (training dataset) analysis and relation to the target variable. The EDA report is generated during model training before the Preprocessing stage. Given the potentially wide feature space, up to 20 of the most important features with the highest statistical significance are selected for EDA based on machine learning modeling. The EDA tool is available on the Training tab found via the **Analytic's Tools** button (See, **2.Train your model** section).

NOTE:

The final model might indicate a slightly different set of features having the highest importance due to a comprehensive number of iterations during training. This could result in discrepancies between the features selected for EDA and the final Feature Importance Matrix.

In the "**Exploratory Data Analysis**" tool you can find the specified information on graphics in each of the following sections:

- **Dataset overview**
This section displays brief data statistics of your training dataset and provides the following information: problem type, dataset dimension and missing values records number.
- **Continuous data distribution and relation to the target variable.**
Visualization of each continuous variable yields two plots:
 - Variable density distribution chart**
A Density plot visualizes the distribution of data across all rows in the dataset. This chart is a variation of a Histogram that uses kernel smoothing to plot values, allowing for smoother distributions by smoothing out the noise. The peaks of a Density plot help display where values are concentrated over the interval.
 - Feature relation to the target variable (different for regression and classification task types)**
This chart is presented in one of the following two formats: line chart, indicating the continuous variable changes with the changes in the continuous target

variable (regression task type) or histogram, showing the mean continuous variable value for each of the classes of the target variable (classification task type).

- **Discrete data distribution and relation to the target variable**

Visualization of each categorical variable yields two plots:

Histogram displaying feature categories count;

Feature categories relation to the target variable (different for regression and classification task types)

This chart is presented in one of two formats, depending on task type: a histogram displaying the mean target variable for each of the feature categories (regression task type) or a histogram displaying the number of each of the target classes in each of the feature classes (classification task type).

- **Feature correlations**

Visualization of the correlations in the data yields two plots:

Heatmap displaying the binary correlation of the 10 most important variables between each other and with the target variable (the 10 most important features are selected based on the binary correlation of the features with the target variable);

Histogram (horizontal) displaying the level of high mutual correlation between independent variable pairs.

Pairs are selected if the value of their mutual correlation exceeds 0.7.

- **Target variable distribution**

Visualization of the target variable statistics is presented in one of the two formats:

Violin plot displaying the distribution, median and outliers in the target variable (regression task type);

Histogram/count plot displaying the number and percentage of each of the target classes throughout the whole dataset (classification task type)

- **Outliers**

Visualization of the outliers in the data is presented in one of the two plots:

Scatter plot displaying the variable distribution in relation to the target variable (regression task type);

Box plot displaying the variable distribution/quantiles/median and the outliers (classification task type).

Outliers are marked purple according to the plots legends.

- **Time Dependencies**

A Time dependency plot is created if a date-time type column is presented in the data. Visualization of time dependency yields three plots each displaying a line chart of the target variable changes over time.

The difference between the charts is the level of data aggregation:

Chart 1: No aggregation. Target variable value is plotted against each date point in the data.

Charts 2 and 3 will dynamically aggregate data into years/months/weeks/days/ hours/minutes.

Aggregation options are automatically selected based on the data timeframe.

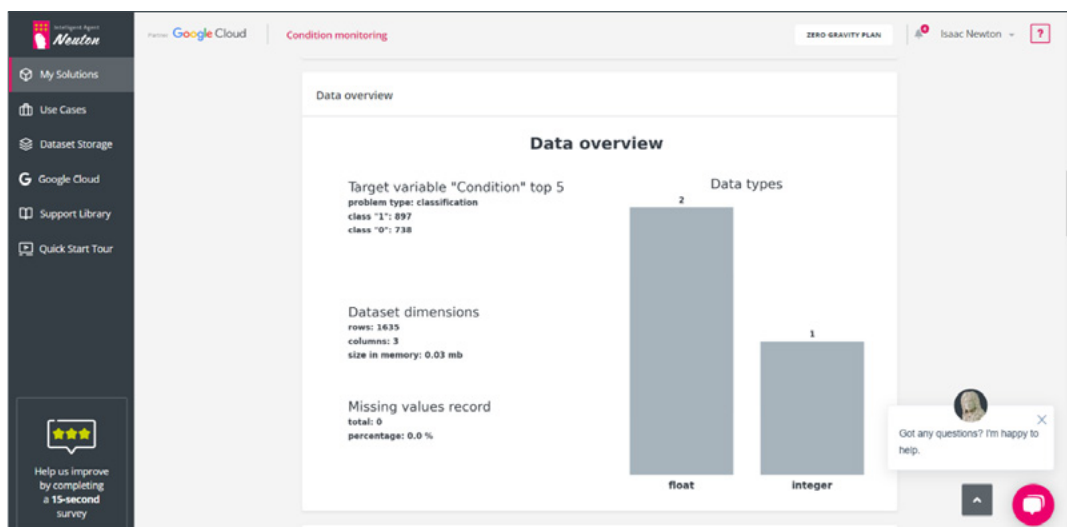
- **Missing Values**

Visualization of the missing values in the data yields two histograms displaying each data feature as an equal bar with missing values indication as against corresponding data indexes.

The **"Missing Values Map Overall Dataset"** plot displays all the data feature bars without feature names and with missing values percentage indication. The purpose of this plot is to give an overall visual representation of the missing values in the data.

The **"Missing Values Map and percentage"** plot displays only the columns which contain missing values with feature names, missing values percentage and the corresponding locations of the missing values in the dataset.

For every dataset the set of graphs is unique. You can also access EDA from the **"My Solutions"** view using the **"Analytic's Tools"** button.



Picture 42 – Exploratory Data Analysis

Model Quality Index

The **Model Quality Index** and **Model Quality Diagram** are the next explainability features inside the Neutron model pipeline and they help users understand the model quality during the training process, and the quality of the trained model.

Model Quality Index determines the quality of the model based on the metric indicator values.

Model Quality Index Value Range: 1 - 100%

Minimum quality: 1%

Maximum quality: 100%

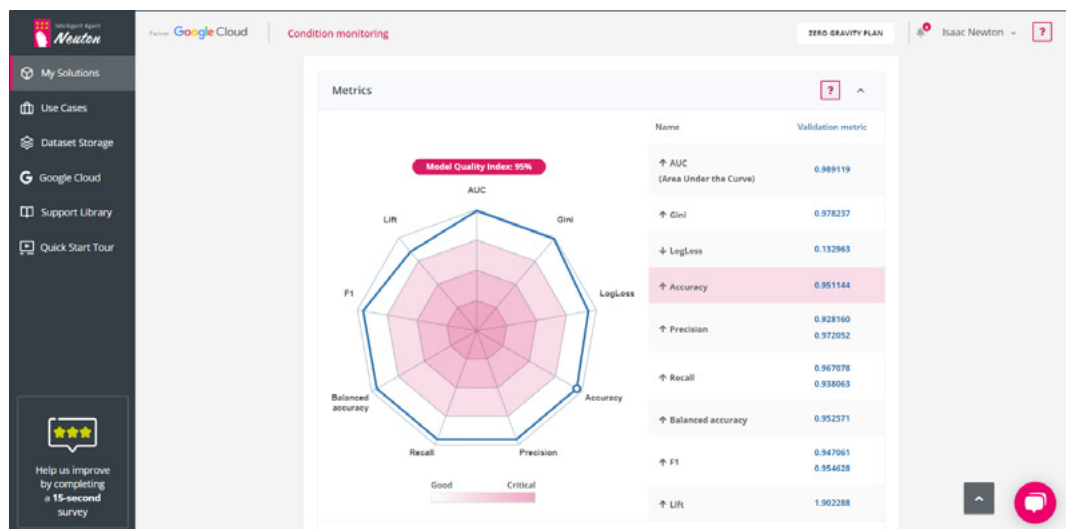
The **Model Quality Index** is calculated during the model training, based on the training or the validation dataset (if the User uploaded the validation dataset for model training). The correlation between the **Training Model Quality Index** and the acceptable model predictive power depends significantly on the problem being solved by the model.

Thus, for tasks that do not require high model predictive power, the acceptable range of the **Model Quality Index** values is 75-100%, while for high-precision tasks it is 99-100%.

The **Model Quality Index** is the aggregated value for the metric indicator values. For more detailed information about metric indicator values please see the **Model Quality Diagram**.

Model Quality Diagram

For any task type you can find the **Model Quality Diagram** to the left of the list of calculated metrics. The **Model Quality Diagram** simplifies the process of evaluating the quality of the model, and also allows users to look at the model from the perspective of various metrics simultaneously in a graphical view. For more details on how to access the list of calculated model metrics, please refer to the "**2. Train your Model. Metrics**" or "**3. Make Predictions**" sections.



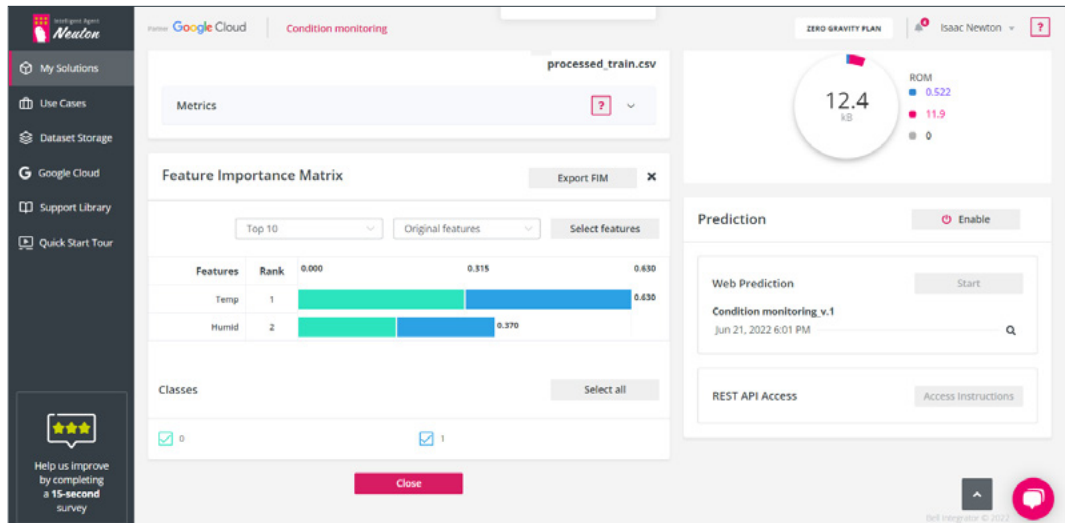
Picture 43 – Model Quality Diagram

A **Model Quality Diagram** is a graphical representation of model quality in relation to metric indicator values that are scaled in the range [0-1], where 1 is the ideal quality of the model, and 0 is the minimum quality of the model. It also allows users to understand metric balance. When the figure displayed is close to the shape of a regular polygon, that conveys a perfect balance between all metric indicator values.

Feature Importance Matrix (FIM)

After the model has been trained, the platform displays a chart with the 10 features that had the most significant impact on the model prediction of the target variable. FIM is available only for solutions with a disabled DSP option. In the **Features** column, you can see the names of features, in the **Rank** column, you can see their importance on model prediction (where the most important feature has rank 1) and on the bar chart you can see the normalized importance value. FIM is available on the Training tab on **Analytic's Tools** button (See, "**2.Train your model**" section). FIM may help you to analyze which features are the most important for the model. To decrease the model total footprint, you can try to train model on the training dataset without unimportant features (features which influence is near 0).

You can also access **FIM** from the "**Prediction**" tab using the "**Analytic's Tools**" button.



Picture 44 – Feature Importance Matrix

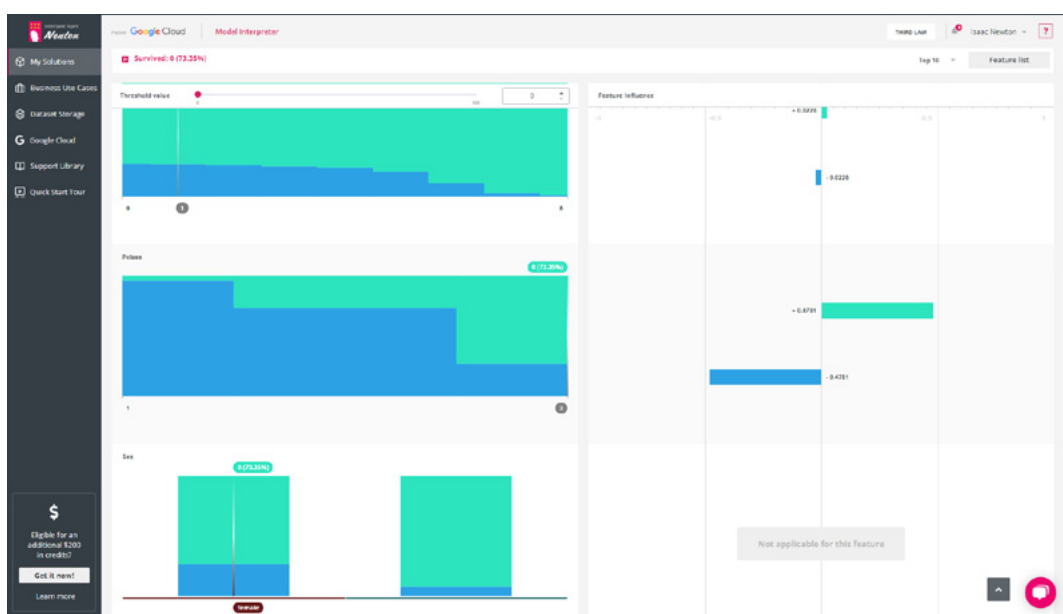
FIM has the following control options:

- **Top 10/Bottom 10**
This control option allows you to show the 10 most important or least important features.
- **Original features/ Original + features after feature engineering**
This option allows you to control what features to display (original features from the training dataset or original features and features after feature engineering).
- **Add feature**
The "**Add Feature**" button allows you to select features of interest.
- **Export FIM**
The "**Export FIM**" button allows you to export FIM in JSON format.
- **Classes (for binary and multiclass classification task types only)**
These check boxes allow you to control what classes to display on on a bar chart.

Model Interpreter

When you are viewing prediction results in the web interface you can click any row to activate the **Model Interpreter** (How to make web predictions, please refer to "**3.Make predictions**" section). It allows users to interactively change original feature values and see how the prediction results would have changed. You can also specify a threshold value for the prediction result and see the corresponding values of the original features. For the numeric features you can see Feature Influence Indicator on the right section of the page. This indicator will show how the prediction will be affected when increasing/decreasing the feature value.

To activate the **Model Interpreter**, click the button that appears when hovering over the particular prediction. The **Model Interpreter** will be activated. Default feature values will correspond to the actual input features used for the prediction. You can then change one feature value at a time to see how the prediction would have been affected.



Picture 45 - Model Interpreter

The **Model Interpreter** offers the following controls:

- **"Back to dataset"**
Use this button to return to the full list of prediction results
- **Predicted Result**
Shows the prediction result for the selected values of the original features (with the predicted class probability for the classification task type).
- **Threshold Value Slider**
Using the slider (or typing manually) you can specify the threshold value of the target variable and Neutron will highlight the corresponding values of the original features for which the prediction result is above or below the threshold value. For the classification task input the "Minimal probability" of the predicted class is used as a threshold. This way you can quickly see which feature values you need as input to predict above the selected threshold.

- **List of features for analysis**

Features are ordered according to their rank in the Feature Importance Matrix.

The categorical features are represented as bar charts. On the horizontal axis you can select any feature value (available feature values derive from the training dataset feature variations). This means that if your test dataset feature has additional categories that the model hasn't seen, those categories will not be available in the **Model Interpreter**. For each categorical feature, you can see the predicted target value for all the feature categories. For the classification task type you can see the predicted class along with the probability.

The numeric features are represented as area charts. The feature values on the horizontal axis are ordered from min to max. All the available feature space (from min to max) is divided into 100 sections. For each numeric feature value you can see the corresponding prediction value.

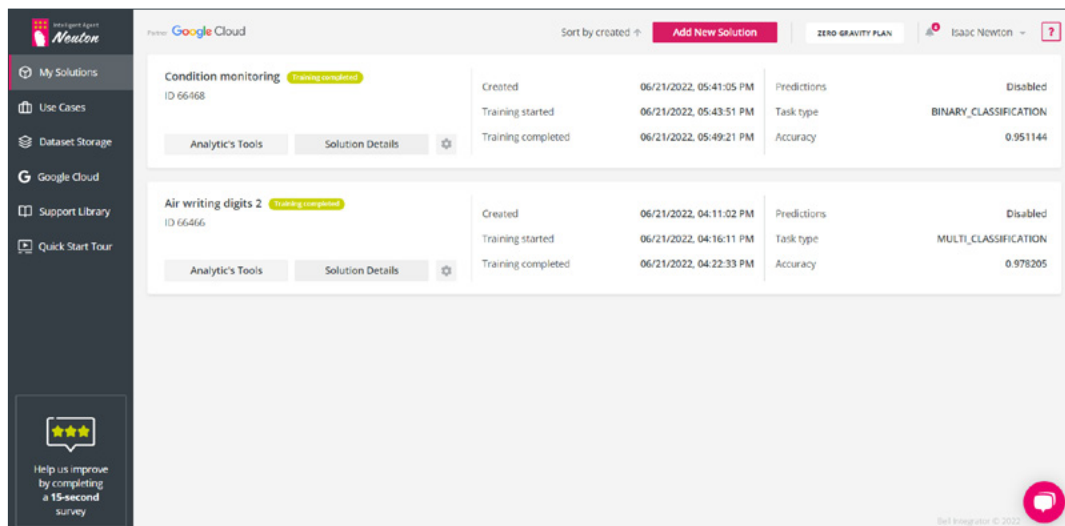
For all features, hovering over any feature value will show you the corresponding prediction details. Also, for any feature, if you left-click and move the pointer to the desired feature value and release the left mouse button, the prediction result will be recalculated given the new feature input. You can click the **Return** button to return to the original feature value.

- **Feature influence**

For the numeric features you can see the feature influence on the prediction result. A positive feature influence value means the predicted target value will increase when the feature value is increased, a negative feature influence value means the predicted target value will decrease if the feature value is decreased.

SOLUTIONS MANAGEMENT

All of your solutions are described in the **"My Solutions"** list in your workspace sorted by creation date in descending order (with the newest first). The solutions creation process is described in the **"1.Select data for training"** section.



Picture 46 - My Solutions

You can change the sorting order by clicking the row near **"Sort by created"**.

In the information area you can see the following information about each respective solution:

- **Created**
date and time when a solution was created
- **Training started**
date and time when model started training
- **Training completed**
date and time when the model completed training
- **Predictions/ Predictions enabled -**
predictions status (disabled by default) or date and time when predictions were enabled
- **Task type**
problem type for which solution was configured (Regression, Binary Classification, Multiclass Classification)
- **Target metric**
selected target metric for the model and its value
- **Solution status** (next to the solution name) reflects the current solution state:
 - **Created**
if you specified a name for your solution and saved it

- **Dataset configured**
if solution was created and dataset was specified
- **Training in progress**
if model is in the training process
- **Training completed**
if model was successfully created
- **Prediction is being enabled**
if you initialized Web Predictions
- **Prediction configured**
if virtual machine for predictions was created

In the control area you can find the following control buttons and indicators:

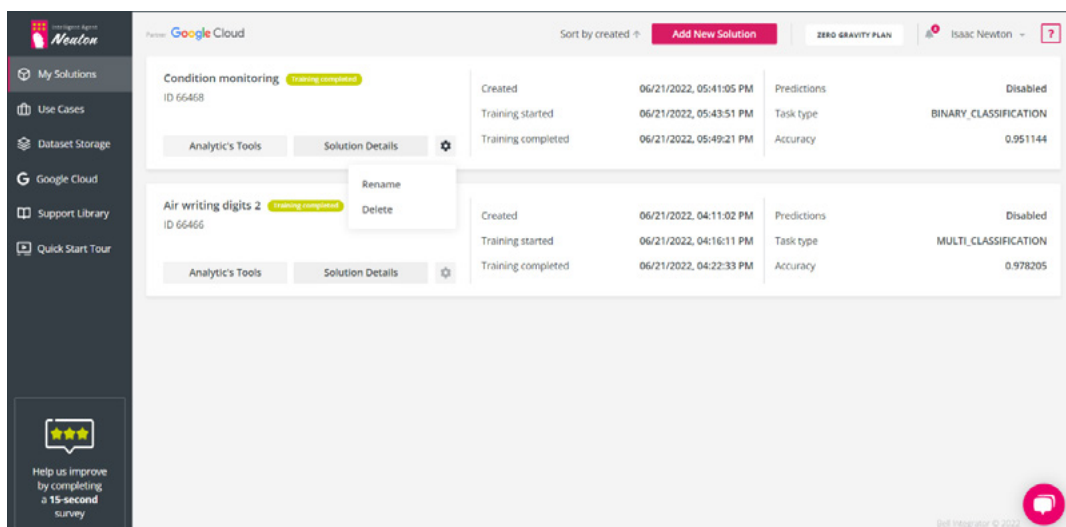
- **Solution Details**
By clicking this option, you will be redirected to the screen with the current solution stage: training dataset selection, training parameters specification, training progress or prediction.
- **Settings menu**
By clicking the gear icon on any solution user can **rename** the solution or **delete** it.
- **Analytic's Tools**
This button opens the EDA tool and FIM menu (See "**Explainability Office**" section).

Solution_ID

You can find the **Solution_ID** under the solution name. For more information about using the **Solution_ID** please refer to the "**Dataset Storage**" section.

Rename Solution

To **rename** a solution, click the **gear** icon on the solution you would like to rename, then click "**Rename**":



Picture 47 - Solution Details gear icon menu

Specify the new **"Solution name"** and/or **"Description"**, then press **"Next"**. The solution will be renamed and/or the description will be updated.

Delete Solution

Click the **gear** icon associated with the solution you would like to delete, then click **"Delete"**. The platform will ask you to confirm that you wish to permanently delete the solution and all associated data in **"Dataset Storage"**. **Once confirmed this cannot be undone.**

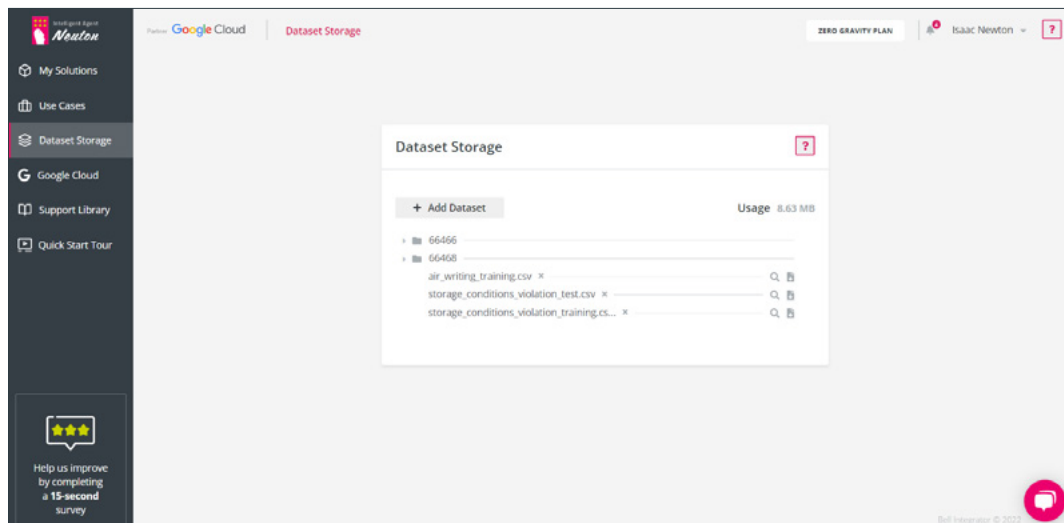
NOTE:

If the prediction service was enabled, the system will warn that in order to delete this solution user must first undeploy the model.

DATASET STORAGE

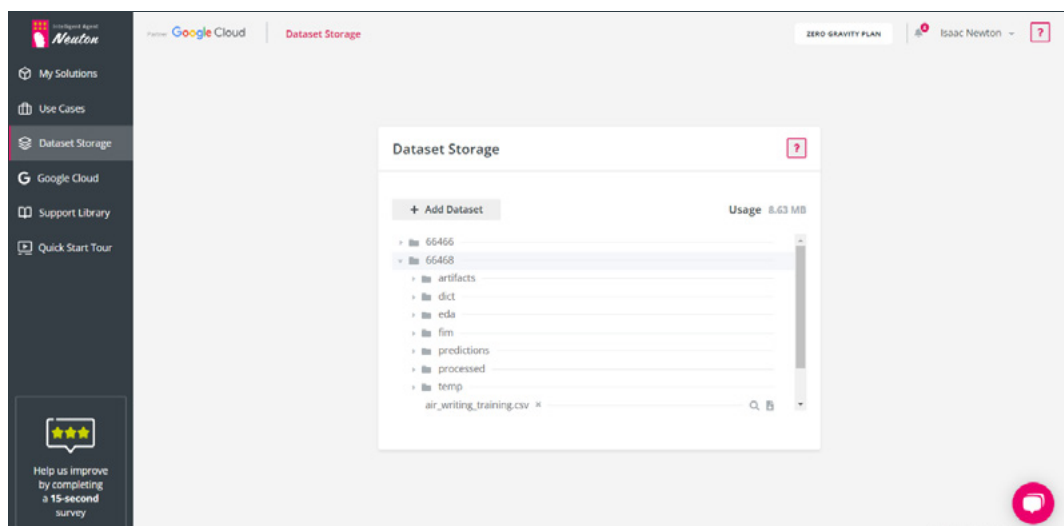
You can manage your data, models, solutions and prediction results via the "**Dataset Storage**" tab.

Click the "**Dataset Storage**" tab on the main screen to access the folder list. Neuton will show the dataset storage screen:



Picture 48 – Dataset Storage

By default "**Dataset Storage**" is empty and populates with new folders during your work in Neuton. You can expand any folder to view the data located inside by clicking on the **folder icon**:



Picture 49 – Expanded view

NOTE:

You can expand any folder by clicking the **right arrow** near the folder icon. In Dataset Storage you can see the list of solutions folders, numbered in consecutive order with a **Solution_ID** (**Solution_ID** is a unique solution identifier used in Neuton).

You can find the **Solution_ID** under the solution name on the "My Solutions" page.

The [**Solution_ID**] folder has the following structure:

- **artifact**
stores downloadable version of a solution
- **dict**
stores dictionaries created by Neuton for model preparation and feature engineering. These dictionaries are used during model training and prediction with the trained model.
- **eda**
stores all information for EDA visualization
- **fim**
stores Feature Importance Matrix for solution
- **predictions**
stores solution web predictions you have made. Subfolders are named after the corresponding solution name. Subfolders contain csv files with the predictions. File names inside of subfolders are formed via the following mask: [**predict_<epochtime>**]. You can **view** prediction results using the "**lens**" icon
- **processed**
stores the processed dataset used for the model training
- **temp**
stores files with service information about the model. These files store model metrics, neural network parameters and metadata

NOTE:

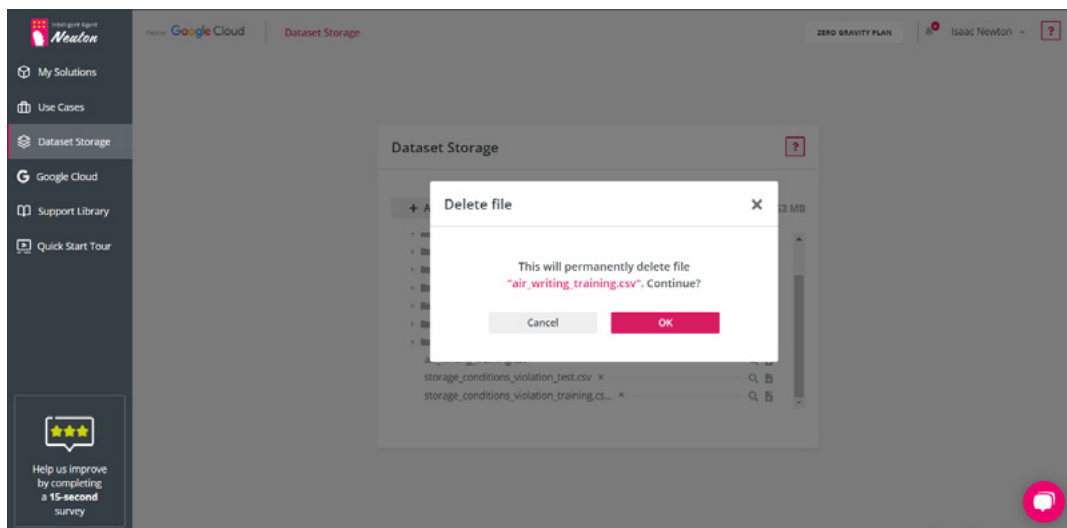
Within the dataset management tab, you can monitor your storage size usage. For example, "Usage 3.23 GB" means that 3.23 GB of Neuton storage is used for your data, models, prediction results, etc.

Dataset Management

Datasets can be managed with the following actions:

- **delete dataset**
- **view dataset**
- **download dataset**

For example, to permanently **delete dataset**, you can use delete icon right next to dataset name. The system will ask you if are you sure to delete it:



Picture 50 – Dataset Management

To **download a dataset**, use the **download icon** right next to **lens icon** which is used for dataset review (described in the "**Select data (Dataset tab)**" section).

METRICS DEFINITION

This section describes the metrics definition for each task type.

Regression Metrics

Mean Absolute Error (MAE)

Mean Absolute Error is the difference between all the observed and predicted values. The direction of the error (positive or negative) does not matter, because the size of the error calculated by module.

Use this metric to minimize the average error.

There are two other metrics dependent on Mean Absolute Error:

- **Max AE** is the maximum absolute difference between the actual value (true) and the predicted value;
- **Min AE** is the minimum absolute difference between the actual value (true) and the predicted value.

For more information, please see https://en.wikipedia.org/wiki/Mean_absolute_error

Root Mean Squared Error (RMSE)

Root Mean Squared Error or RMSE represents an error between observed and predicted values (square root of squared average error over all observations). The lower RMSE – the better the models predictive power.

RMSE is always non-negative, and a value of 0 would indicate a perfect fit for the data.

It should be used when you want to make a model which would not have big individual errors for every prediction.

For more information, please see https://en.wikipedia.org/wiki/Root-mean-square_deviation

Root Mean Squared Logarithmic Error (RMSLE)

Root Mean Squared Logarithmic Error or RMSLE can be used when one doesn't want to penalize huge differences when both the values are huge numbers. The lower RMSLE – the better the models predictive power.

RMSLE can also be used when one wants to penalize underestimates more than overestimates.

For more information, please see https://en.wikipedia.org/wiki/Root-mean-square_deviation

Coefficient of Determination (R2)

Coefficient of Determination is the proportion of the variance in the dependent variable that is predictable from the independent variables. This metric scores the model 1 if our model is perfect, 0 otherwise.

This means if Coefficient of Determination = 0.95 – then 95% of data is explained by observed statistics and thus by the trained model.

For more information, please see https://en.wikipedia.org/wiki/Coefficient_of_determination

Mean Squared Error (MSE)

MSE is a measure of the quality of an estimator, it is always non-negative, and values closer to zero are better.

MSE measures the average of the squares of the errors - that is, the average squared difference between the estimated values and true values.

The squaring is necessary to remove any negative signs, it also gives more weight to larger differences, so bigger errors are penalized higher.

Please follow the link for more details https://en.wikipedia.org/wiki/Mean_squared_error

Root Mean Squared Percentage Error (RMSPE)

Root Mean Squared Percentage Error represents a percentage error between the observed and predicted values measured as the square root of the mean of the squared (difference between the actual values and the predicted values divided by the actual values). The lower the RMSPE – the better the model's predictive power.

RMSPE is always non-negative, and a value of 0 (almost never achieved in practice) would indicate a perfect fit to the data. In general, a lower RMSE is better than a higher one. However, comparisons across different types of data would be invalid because the measure is dependent on the scale of the numbers used.

Note: rows with 0 values in the target variable are filtered out and are not used in the validation pipeline as division by 0 is not possible.

Please follow the link for more details: https://en.wikipedia.org/wiki/Mean_percentage_error

Binary classification metrics

Accuracy

Accuracy represents how accurately class is predicted. If from 100 predicted records 73 had been assigned a correct class, then the accuracy will be 0.73 or 73%. The higher value is the better.

Accuracy does not take class imbalance into account. It is best applicable when classes are represented by an approximately equal quantity of samples.

For more information, please see https://en.wikipedia.org/wiki/Accuracy_and_precision

Balanced Accuracy

Balanced Accuracy accounts for imbalanced classes where (for example) one class may be represented by 10% of samples and the second class may be represented by the remaining 90% of samples.

For more information, please see https://en.wikipedia.org/wiki/Precision_and_recall

Precision

Precision is the fraction of relevant samples among the retrieved samples. The precision score reaches its best value at 1 and worst score at 0. Precision is intuitively the ability of the classifier not to label as positive a sample that is negative. Use this metric when the

priority is to find only relevant samples without a mistake, even if you may end up skipping some of the relevant samples.

For more information, please see: https://en.wikipedia.org/wiki/Precision_and_recall

Recall

Recall is the fraction of relevant samples that have been retrieved over the total amount of relevant samples. The recall score reaches its best value at 1 and worst score at 0. Recall is the ability of the classifier to find all the positive samples. Use this metric when the priority is to find as many relevant samples as possible even if you may also mark some of the wrong examples as relevant.

For more information, please see https://en.wikipedia.org/wiki/Precision_and_recall

F1 Score

The F1 Score can be interpreted as a weighted average of the precision and recall for each class, where an F1 Score reaches its best value at 1 and worst score at 0. You should use this metric when you want to have a good balance between Precision and Recall.

For more information, please see: https://en.wikipedia.org/wiki/F1_score

Confusion Matrix

Confusion Matrix, also known as an error matrix, is a specific table layout that allows visualization of the performance of an algorithm. Each row of the Matrix represents the samples in a predicted class while each column represents the samples in an actual class (or vice versa). Matrix makes it easy to see if the system is confusing two classes.

For more information, please see https://en.wikipedia.org/wiki/Confusion_matrix

Gini

Gini coefficient applies to binary classification and requires a classifier that can in some way rank examples according to the likelihood of being in a positive class. A Gini value of 0% means that the characteristic cannot distinguish between classes.

Gini coefficient makes sense for the whole collection of predictions and not individual data points. Gini Coefficient only tells if perfect segregation (based on probability predictions) is possible or not and nothing about the probability threshold. In short, there is no

relation between probability threshold and Gini. Gini coefficient provides accurate model predictive power measure for imbalanced class problems.

For more information, please see https://en.wikipedia.org/wiki/Gini_coefficient

AUC (ROC AUC)

A Receiver Operating Characteristic curve is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied.

Classification metric "Area Under the Curve" (AUC) related to the ROC curve represents how effective the model answers to the question "Does the current object belong to the corresponding class".

The value is always between 0 and 1:

- The higher = the better;
- The lower = the worse, but in this case model works in "reverse" mode (1 - value = the better);
- ~0,5 = the worst (the model provides values like random)

For more information, please see https://en.wikipedia.org/wiki/Receiver_operating_characteristic

Lift

Lift is a measure of the performance of a targeting model at predicting or classifying cases as having an enhanced response (with respect to the population as a whole), measured against a random choice targeting model.

For more information, please see [https://en.wikipedia.org/wiki/Lift_\(data_mining\)](https://en.wikipedia.org/wiki/Lift_(data_mining))

LogLoss

Logarithmic Loss is a measure of prediction confidence level. LogLoss represents the difference between the actual class and the probability of a prediction being in that class. For example, the model correctly predicts a 0.90 probability of being in class 1- that means it is pretty confident, but still there is 0.1 uncertainty of this prediction; LogLoss penalizes for this uncertainty.

The lower the LogLoss score, the better the model's predictive power. LogLoss takes into account not the rounded-off predicted class but the probability of the prediction to correspond to a certain class.

For more information, please see https://en.wikipedia.org/wiki/Loss_functions_for_classification

Multiclass classification metrics

Accuracy

Accuracy represents how accurately class is predicted. If from 100 predicted records 73 have been assigned a correct class, then the accuracy will be 0.73 or 73%. The higher the value is, the better.

Accuracy does not take class imbalance into account. It is most applicable when classes are represented by an approximately equal quantity of samples.

For more information, please see: https://en.wikipedia.org/wiki/Accuracy_and_precision

Balanced Accuracy

Balanced Accuracy accounts for imbalanced classes where (for example) one class may be represented by 10% of samples, the second class may be represented by 60% of samples and the other N classes are represented by the remaining 30% of samples.

For more information, please see https://en.wikipedia.org/wiki/Precision_and_recall

Macro Average Precision

Precision is the fraction of relevant samples among the retrieved samples. The precision score reaches its best value at 1 and worst score at 0. Precision is intuitively the ability of the classifier not to label as positive a sample that is negative. Use this metric when it is the most important to find only relevant samples without mistake even if you may skip some of the relevant samples. Macro Average Precision does not take class imbalance into account. It is best applicable when (for example) three classes of a multiclass classification problem, represented by an approximately equal quantity of samples.

For more information, please see https://en.wikipedia.org/wiki/Precision_and_recall

Weighted Average Precision

Weighted Average Precision accounts for imbalanced classes where (for example) one class may be represented by 10% of samples, the second class may be represented by 60% of samples and the other N classes are represented by the remaining 30% of samples.

For more information, please see https://en.wikipedia.org/wiki/Precision_and_recall

Macro Average Recall

Recall is the fraction of relevant samples that have been retrieved over the total amount of relevant samples. The recall score reaches its best value at 1 and worst score at 0. Recall is intuitively the ability of the classifier to find all the positive samples. Use this metric when it is the most important to find as many relevant samples as possible even if you may also mark some of the wrong examples as relevant. Macro Average Recall does not take class imbalance into account. It is best applicable when (for example) three classes of a multiclass classification problem, represented by an approximately equal quantity of samples.

For more information, please see https://en.wikipedia.org/wiki/Precision_and_recall

Weighted Average Recall

Weighted Average Recall accounts for imbalanced classes where (for example) one class may be represented by 10% of samples, the second class may be represented by 60% of samples and the other N classes are represented by the remaining 30% of samples.

For more information, please see https://en.wikipedia.org/wiki/Precision_and_recall

Macro Average F1 Score

The F1 score can be interpreted as an arithmetic average of the precision and recall for each class, where an F1 score reaches its best value at 1 and worst score at 0. You should use this metric when you want to have a good balance between Precision and Recall. Macro average F1 score does not take class imbalance into account. It is best applicable when (for example) three classes of a multiclass classification problem, represented by an approximately equal quantity of samples.

For more information, please see https://en.wikipedia.org/wiki/F1_score

Weighted Average F1 Score

Weighted Average F1 Score accounts for imbalanced classes where (for example) one class may be represented by 10% of samples, the second class may be represented by 60% of samples and the other N classes are represented by the remaining 30% of samples.

For more information, please see https://en.wikipedia.org/wiki/F1_score

Confusion Matrix

Confusion Matrix, also known as an error matrix, is a specific table layout that allows visualization of the performance of an algorithm. Each row of the Matrix represents the samples in a predicted class while each column represents the samples in an actual class (or vice versa). Matrix makes it easy to see if the system is confusing two classes.

For more information, please see https://en.wikipedia.org/wiki/Confusion_matrix

LogLoss

Logarithmic Loss is a measure of prediction confidence level. LogLoss represents the difference between the actual class and the probability of a prediction being in that class. For example, the model correctly predicts a 0.90 probability of being in class 1- that means it is pretty confident, but still there is 0.1 uncertainty of this prediction; LogLoss penalizes for this uncertainty.

The lower LogLoss score, the better the model's predictive power. LogLoss takes into account not the rounded-off predicted class but the probability of the prediction to correspond to a certain class.

For more information, please see https://en.wikipedia.org/wiki/Loss_functions_for_classification